IDC/AUTOSAINT/SDD
21 May 2008

English only

CTBTO PREPARATORY COMMISSION | preparatory commission for the comprehensive nuclear-test-ban treaty organization

AWST science software technology

AWST-TR-08/10

autoSaint Software Design Description

This document defines the *autoSaint* software design description. The software design includes the architectural design, detailed design and interface descriptions.

Summary

autoSaint is a software system that automatically processes particulate and Xenon noble gas radionuclide data, in order to detect any radionuclide isotopes present in the sample. The software runs automatically without human intervention. It reads processing parameters from the database. It processes the sample data according to the specified parameters and writes the results back to the database. The results can then be analysed further using separate interactive analysis software.

Document History

Version	Date	Author	Description
0.1	1 February 2007	Marian Harustak	Initial draft of the document
1.0	27 March 2007	Marian Harustak	Delivered initial SDD
1.1	3 April 2007	Marian Harustak	Revised version addressing IDC comments
2.0	31 October 2007	Marian Harustak Thierry Ferey	Added descriptions in Scientific Calculations library, updated configuration parameters; modified language to the “as built” situation
2.1	21 May 2008	Marian Harustak	Added description of Xenon parts
2.2	28 April 2011	Marian Harustak	Updated to *autoSaint* version 2.1.3

Contents

1. Scope 5
1.1. Identification 5
1.2. System overview 5
1.3. Document overview 6
2. Software architecture 7
2.1. Software decomposition 8
2.1.1. autoSaint Pipeline Wrapper 8
2.1.2. Scientific Calculations Library 9
2.1.3. Additional Calculations Library 9
2.1.4. Supporting Functions Library 9
2.1.5. Infrastructure Library 9
2.1.6. gODBC 9
2.2. Rationale 9
2.3. General Implementation 10
2.3.1. Requirements 10
2.3.2. Design decisions 10
3. Processing entities 12
3.1. *autoSaint* Pipeline Wrapper 12
3.1.1. Overview 12
3.1.2. Dependencies 12
3.1.3. Requirements 12
3.1.4. Design decisions 14
3.2. Scientific Calculations Library 19
3.2.1. Overview 19
3.2.2. Dependencies 19
3.2.3. Requirements 20
3.2.4. Design decisions 20
3.3. Additional Calculations Library 28
3.3.1. Overview 28
3.3.2. Dependencies 28
3.3.3. Requirements 28
3.3.4. Design decisions 28
3.4. Supporting Functions Library 43

3.4.1. Overview 43
3.4.2. Dependencies 43
3.4.3. Requirements 43
3.4.4. Design decisions 43
3.5. Infrastructure Library 50
3.5.1. Overview 50
3.5.2. Dependencies 50
3.5.3. Requirements 50
3.5.4. Design decisions 51
4. Interface entities 53
4.1. Data Access 53
4.1.1. Overview 53
4.1.2. Dependencies 53
4.1.3. Requirements 53
4.1.4. Design Decisions 53
Appendix I Additional Requirements 55
Appendix II CONFIGURATION PARAMETERS 58
Appendix III PARTICULATES Processing Sequence as defined in tor and as Implemented 64
XENON Processing Sequence as defined in tor and as implemented 66
Appendix IV Abbreviations 68
References 69

1. SCOPE

1.1. Identification

This document applies to the *autoSaint* version 2.1.3.

1.2. System overview

The IMS (International Monitoring System) includes among others also radionuclide stations, where particulate and noble gas monitoring systems are installed. These systems send spectrum data to the IDC (International Data Centre) in Vienna on a daily basis. The IDC processes and reviews the spectrum data. Analysis is performed in two separate pipelines: The automatic pipeline where each incoming spectrum is processed automatically and the manual pipeline where the same spectrum and its automatic analysis are reviewed by a radionuclide analyst. Both processes produce analysis reports, which conform to a specified IDC format.

The *autoSaint* software automatically processes gamma spectral data from particulate stations equipped with HPGe detectors and noble gas stations equipped with SPALAX detectors. This processing will occur after the data have been parsed and before the Automatic Radionuclide Report (ARR) is produced.

For each received spectrum, the software calibrates the spectral data using the latest calibration pairs (resolution, energy and efficiency) and finds the reference peaks defined in the database. The calibration routine consists in calculating the spectrum baseline, the SCAC and LC calculation, and then fine tuning the peak characteristics for each peak found.

After calibrating the spectrum and updating the calibration pairs, the processing diverges for particulate and xenon noble gas samples.

- For a particulate sample, the peak finding process is being repeated to recalculate the three described quantities (energy, resolution, efficiency). In this phase the new Spectrum Baseline, the SCAC and the peaks found are stored. As the next step, the Nuclide Identification Routine runs using the last efficiency calibration.
- For noble gas sample, a xenon analysis routine is executed. In this phase the new Spectrum Baseline, the SCAC and the characteristics of four xenon isotopes are calculated.

Afterwards, for both particulates and xenon, the software calculates the activity concentration and the MDC for the energy of interest. The results of the processing are stored in the database and file store.

The software also runs the Quality Control program with the results being stored in the database.

1.3. Document overview

This document defines the *autoSaint* version 2.1.3 software design. The software design includes the architectural design, detailed design and interface descriptions.

This document is mainly intended for developers, maintainers and documentation writers. It is also of interest to project management, requirements analysts, quality assurance staff and user representatives.

The design is described in terms of a set of connected entities. An entity is an element (component) that is structurally and functionally distinct from other elements and that is separately named and referenced. Entities may be sub-systems, data stores, modules, programs, processes, or object classes. Entities may be nested or form hierarchies.

Each entity is described in terms of requirements and design decisions.

Each mandatory, testable requirement is stated using the word **shall**. Therefore, each **shall** in this document should be traceable to a documented test. Each mandatory, non-testable requirement is stated using the word **will**. Each recommended requirement is stated using the word **should**. A permissible course of action is stated using the word **may**. This convention is used in ISO/IEC 12207.

Each mandatory, design decision is stated using the word **will**. Each design recommendation is stated using the word **should**. A permissible course of action is stated using the word **may**. This convention is used in ISO/IEC 12207.

This document is compliant with the IDC Software Documentation Framework (2002) and the CTBTO Editorial Manual (2002).

2. SOFTWARE ARCHITECTURE

The architectural decomposition of the *autoSaint* software is shown in Figure 1.

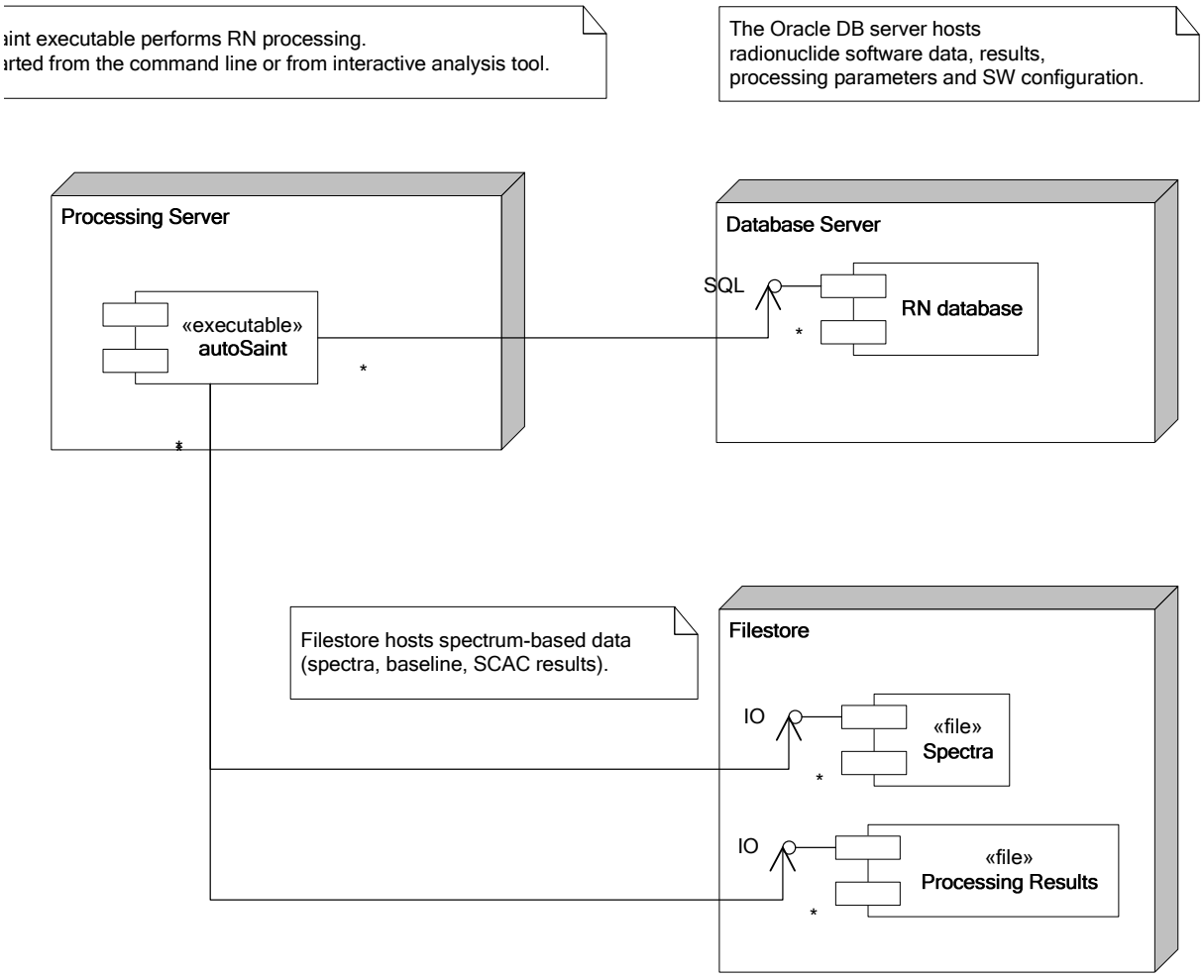

Figure 1 - *autoSaint* components

The main component is the *autoSaint* executable. This executable contains all of the functionality described in this SDD. It accesses the data stored on the relational database server (Oracle SQL) and on the file based data store (e.g. spectra files). The relational data store holds the software configuration, processing parameters, part of the input data and processing results. The file based data store holds the spectra based data (input spectra, baselines, SCACs).

2.1. Software decomposition

The architectural software decomposition of the *autoSaint* software is shown in Figure 2.

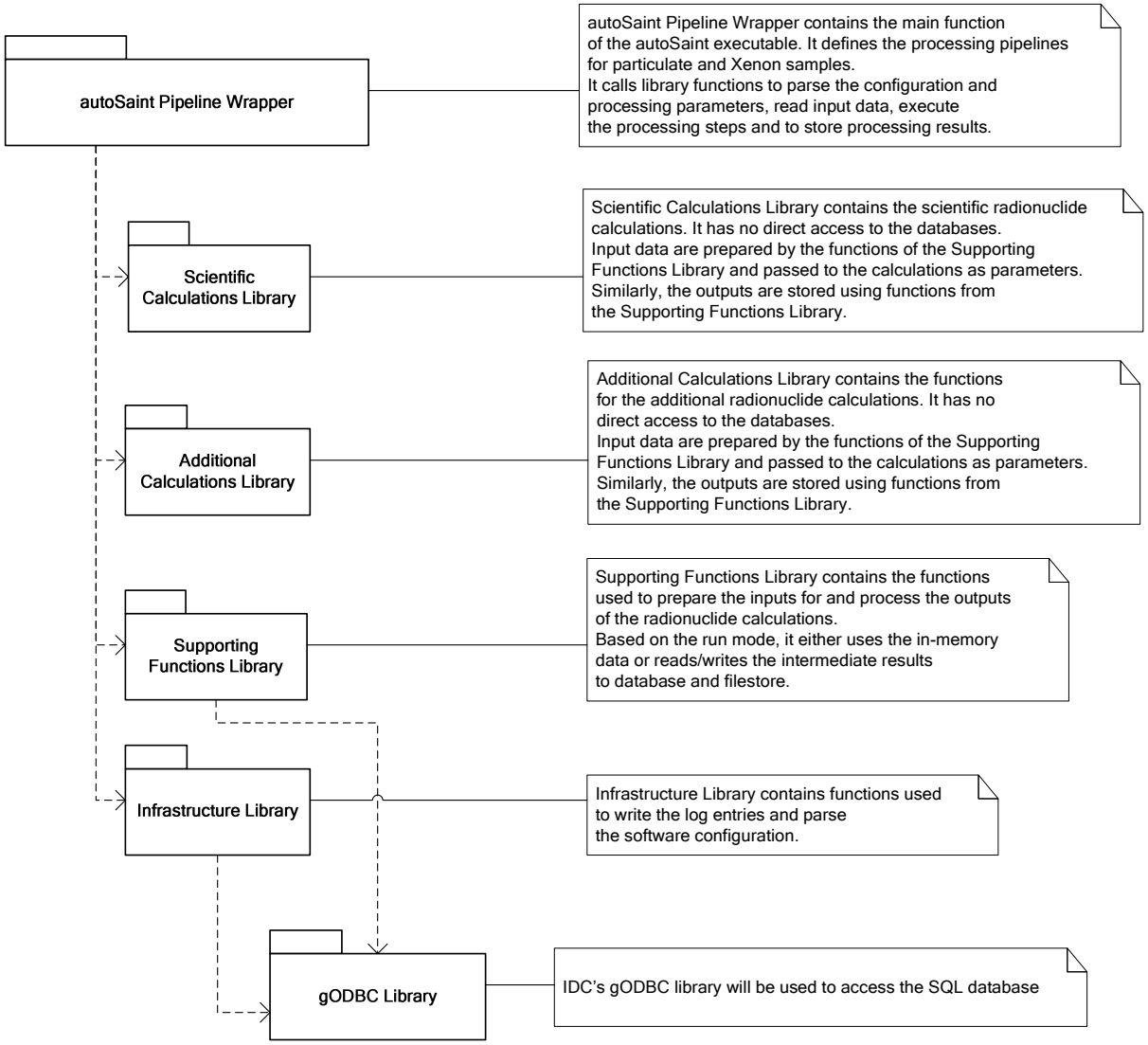
autoSaint Pipeline Wrapper
autoSaint Pipeline Wrapper contains the main function of the autoSaint executable. It defines the processing pipelines for particulate and Xenon samples. It calls library functions to parse the configuration and processing parameters, read input data, execute the processing steps and to store processing results.
Scientific Calculations Library
Scientific Calculations Library contains the scientific radionuclide calculations. It has no direct access to the databases. Input data are prepared by the functions of the Supporting Functions Library and passed to the calculations as parameters. Similarly, the outputs are stored using functions from the Supporting Functions Library.
Additional Calculations Library
Additional Calculations Library contains the functions for the additional radionuclide calculations. It has no direct access to the databases. Input data are prepared by the functions of the Supporting Functions Library and passed to the calculations as parameters. Similarly, the outputs are stored using functions from the Supporting Functions Library.
Supporting Functions Library
Supporting Functions Library contains the functions used to prepare the inputs for and process the outputs of the radionuclide calculations. Based on the run mode, it either uses the in-memory data or reads/writes the intermediate results to database and filestore.
Infrastructure Library
Infrastructure Library contains functions used to write the log entries and parse the software configuration.
gODBC Library
IDC's gODBC library will be used to access the SQL database

Figure 2 – Software decomposition

The *autoSaint* software can be decomposed into the Pipeline Wrapper and various libraries. The Pipeline Wrapper contains the top-level logic and calls other library functions to perform various activities. The rationale behind the decomposition is to make the software modular on the source code level and to facilitate later modifications of the processing pipeline.

2.1.1. autoSaint Pipeline Wrapper

The Pipeline Wrapper is the top-level executable component of the software. It executes the complete automatic pipeline.

The Pipeline Wrapper executes the default processing sequence for particulate or Spalax sample as defined in the Terms of Reference (TOR) and in subsequent meetings with the

CTBTO representatives. The description of the processing sequence is included in Appendix III.

The integrity of the database and log files is ensured by using the sample ID in all database and log file entries. The sample ID thus serves as a cross-data store logical key.

2.1.2. Scientific Calculations Library

The Scientific Calculations Library contains all scientific functionality of the software, such as baseline, SCAC and LC calculations, peak search, nuclide identification and Xenon analysis. The details of the scientific calculations are described in section 3.2.

2.1.3. Additional Calculations Library

The Additional Calculation Library contains the additional calculation functions needed to perform the processing pipeline, like activity and MDC calculation and application of the QC algorithm. The details of the additional calculations are described in section 3.2.4.6.

2.1.4. Supporting Functions Library

The Supporting Functions Library is responsible for preparing the data for the calculation routines and for parsing the results. The details of the supporting functions are described in section 3.4.

2.1.5. Infrastructure Library

The Infrastructure Library contains the infrastructure functions like reading the software configuration and writing log entries. The details of the infrastructure functions are described in section 3.4.4.8.

2.1.6. gODBC

The IDC's gODBC library is used to access the SQL database. The gODBC library is not a part of the *autoSaint* software. Details of the data access are described in section 4.1.

2.2. Rationale

The rationale behind the software decomposition as described in section 2.1 is to provide the software with a high degree of configurability on the run-time level (for example allowing the users to reprocess a sample using different parameters) and a high degree of modularity on the source code level. The individual steps performed in the Pipeline Wrapper are largely independent from a source code point of view. This approach allows for an easier integration of additional processing steps.

The decomposition of the software into multiple components based on their functions also improves the maintainability of the software by making the source code easier to read and navigate.

2.3. General Implementation

The general requirements affecting the design of the *autoSaint* software, which are not specified by the detailed design description, are listed in the section 2.3.1 and are addressed in the section 2.3.2.

2.3.1. Requirements

General implementation requirements affecting the architecture, as specified in [AUTO_SAINT_SRS] and [AUTO_XE_SAINT_SRS]:

1. The software shall be implemented in ANSI C.
2. It shall be possible to regenerate all executables using GNU auto-tools.
3. The software shall compile correctly (without warnings) with both the Sun workshop compiler (version 6.2 or higher) and the GNU C compiler (version 3.4.0 or higher). *Note: Sun workshop compiler compatibility is no longer required.*
4. The software shall compile correctly with the GNU C compiler on both Solaris and Linux platforms. *Note: Solaris compatibility is no longer required.*
5. The source code shall meet the requirements specified in the [IDC_CS_2002].
6. The software shall be written in a modular fashion, so as to be extendable and to allow alternative calculation methods to be added.
7. The software shall be able to execute and completely meet all requirements on a Sun Blade 1500 sparc or better, 1Gb RAM, running Solaris (version 9 or later). *Note: Sun Solaris compatibility is no longer required.*
8. The software shall be able to execute, completely meeting all requirements, on a 1.7 GHz Pentium-4 processor with 256 MB of RAM, running Linux (Red Hat 4.2 or later).
9. The system should interface with the existing tables in the database wherever possible. This is because changing the database may impact other systems.

Note: Additional requirements were identified and corresponding software changes designed and implemented in the test use of autoSaint. These requirements were recorded in AWST Jira issue tracking tool.

2.3.2. Design decisions

Design decisions addressing the general implementation requirements:

1. The software was implemented in ANSI C. The design described in this document reflects this design decision.
2. The GNU auto-tools were used during the development of the software.
3. The software is built so that it compiles correctly (without warnings) with both the Sun workshop compiler (version 6.2 or higher) and the GNU C compiler (version 3.4.0 or higher). *Note: Sun workshop compiler compatibility is no longer required.*
4. The software is built so that it compiles correctly with the GNU C compiler on both Solaris and Linux platforms. *Note: Solaris compatibility is no longer required.*

5. The software was coded in compliance with the coding standard [IDC_CS_2002].The modularity of the software is described in the section 2.
6. The software was written for and tested on a Sun Blade 1500 sparc or better, 1Gb RAM, running Solaris (version 9 or later). *Note: Sun Solaris compatibility is no longer required.*
7. The software was written for and tested on host running a Linux Red Hat 4.2 or later.
8. The software design and implementation minimizes the need for changes of the existing tables in the database. This is because changing the database may impact other systems.

3. PROCESSING ENTITIES

3.1. *autoSaint* Pipeline Wrapper

3.1.1. Overview

The Pipeline Wrapper is the core of the *autoSaint* executable, the top-level executable component of the software. It is used to either execute a complete automatic pipeline or only an individual step.

In terms of functionality, the Pipeline Wrapper contains only the pipeline logic. All scientific calculations, supporting and infrastructure functions are implemented in the libraries.

3.1.2. Dependencies

The Pipeline Wrapper depends on all other libraries of the *autoSaint* software, namely the Scientific Calculations Library, Additional Calculations Library, Supporting Function Library and Infrastructure Library.

It also depends on both data stores: the file based data store and the relation database.

It uses the library interfaces defined in the corresponding library header files.

3.1.3. Requirements

Table 1- Requirements allocated to Pipeline Wrapper

Requirement	Addressed by
The software shall be able to run completely automatically without any operator intervention.	The design of the Pipeline Wrapper and the handling of configuration.
The user shall have full access to the software's functionality without needing a GUI or any other interface software.	The design of Pipeline Wrapper.
The software shall be able to process 20 samples simultaneously, with each sample being processed with different parameters. It shall be possible to automatically process 20 sets of sample data simultaneously	Each sample can be processed by a different instance of the *autoSaint*. There is no limitation on the number of *autoSaint* instances running in parallel apart from those imposed by the operating system and/or database.
The software shall require a user ID and password before starting the automated processing.	User access control is performed using the database login credentials defined in the software configuration or on a command line. See section 3.1.4.2.1

Requirement	Addressed by
The software shall have the capability to read the password from a file.	User access control is performed using the database login credentials. It is possible to read them from a file. See section 3.1.4.2.1.
The software shall allow the user to specify: (a) Whether the default login or a specific login should be used; (b) The sample IDs to process	(a) Only the DB login is used (covered by set of requirements defining the database login credentials). (b) See section 3.1.4.2.1
If during initialization the current user is the super user then the software shall generate an error and terminate.	See section 3.1.4.2.2
The software shall provide a database login identifier and password when connecting to the database.	See section 3.1.4.2.1
The system shall have the capability to read the database login and password from a file.	See section 3.1.4.2.1
The system shall allow processing parameters to be adapted without recompiling the software.	See section 3.5.4.1
The automatic processing capability shall be able to execute completely and independently of the interactive analysis.	The design of the Pipeline Wrapper.
The software shall be able to run in parallel with the other IDC operational radionuclide software systems without affecting those systems.	The design of the Pipeline Wrapper. The only effect on other systems will be the sharing of hardware and operating system resources, if run on the same host, and sharing of database server resources.
It shall be possible for multiple instances of the software to run on a single platform.	The *autoSaint* software allows for multiple instances to run on a single platform, each processing a different sample (identified by a sample ID).
The software shall be able to log at start-up the values of all configurable values.	See section 3.1.4.2.1
If the software is unable to connect to the database then the software shall generate an error and terminate.	See section 3.1.4.2
If the software is unable to read from the database, any of the parameters required for automatic processing then the software shall generate an error message and terminate.	See section 3.1.4.2

Requirement	Addressed by
The software shall never overwrite the input data or the output data for a particular sample from the automatic processing tables, in the Auto database.	This requirement was changed upon agreement with the customer. The new requirement: The software shall never overwrite the input data from the automatic processing tables. By default, the software shall not overwrite the output data of automatic processing. It shall be possible to override this restriction by the configuration parameter. The new requirement is addressed in section 3.1.4.2.4.

Note: Additional requirements weere identified and corresponding software changes designed and implemented in the test use of autoSaint.

3.1.4. Design decisions

3.1.4.1. Pipeline Architecture

The Pipeline Wrapper processing is described in the flow diagram in Figure 3.

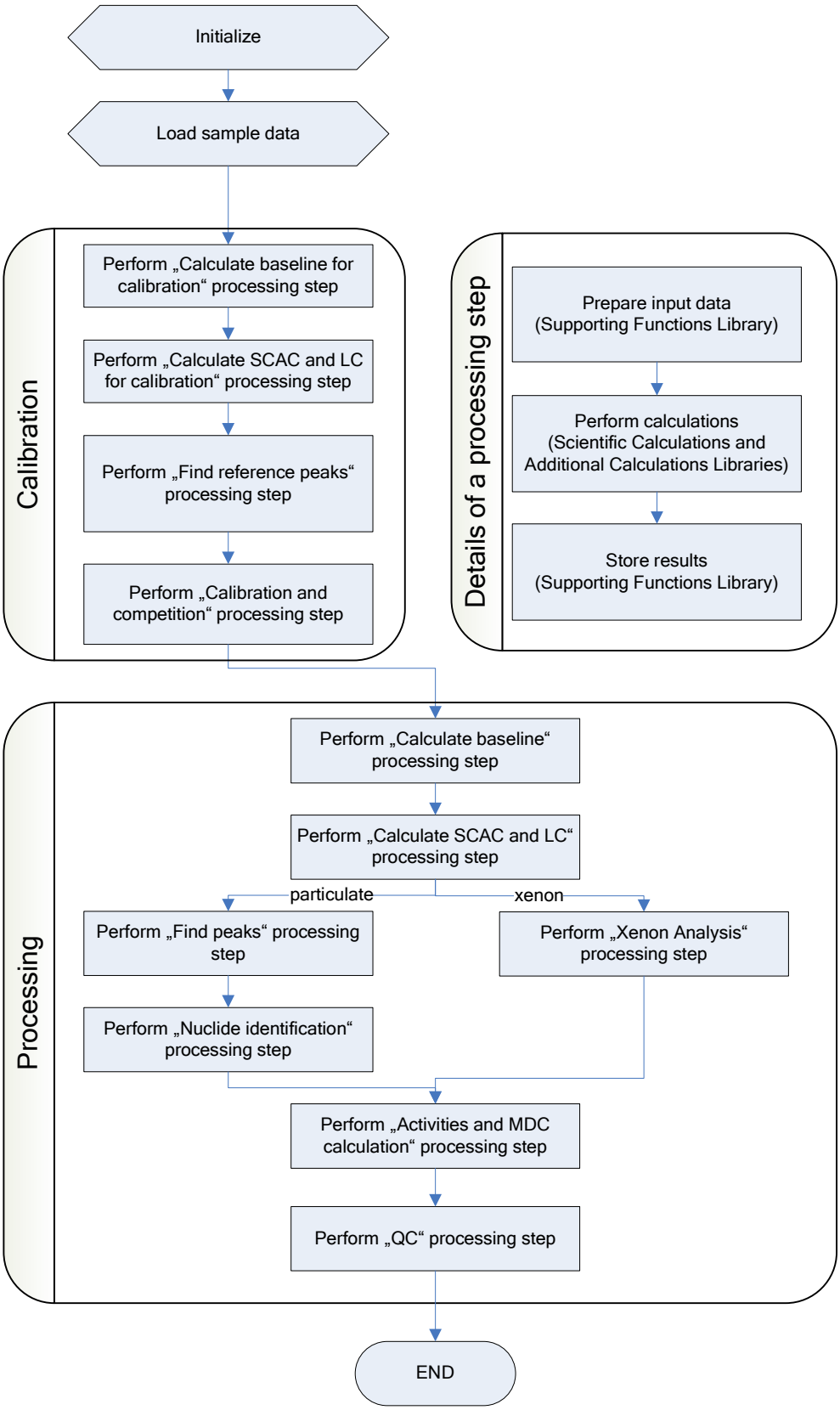
Initialize
Load sample data
Calibration
Perform „Calculate baseline for calibration“ processing step
Perform „Calculate SCAC and LC for calibration“ processing step
Perform „Find reference peaks“ processing step
Perform „Calibration and competition“ processing step
Details of a processing step
Prepare input data (Supporting Functions Library)
Perform calculations (Scientific Calculations and Additional Calculations Libraries)
Store results (Supporting Functions Library)
Processing
Perform „Calculate baseline“ processing step
Perform „Calculate SCAC and LC“ processing step
particulate
xenon
Perform „Find peaks“ processing step
Perform „Xenon Analysis“ processing step
Perform „Nuclide identification“ processing step
Perform „Activities and MDC calculation“ processing step
Perform „QC“ processing step
END

Figure 3 - Pipeline Wrapper processing sequence

Initial steps include the reading of the radionuclide measurement sample from the SQL and file-based data stores, identified by the sample ID which is provided as a command line parameter.

The Pipeline Wrapper performs a sequence of processing steps. This forms the full pipeline process. Intermediate data are passed between individual steps using in-memory data structures.

Each processing step consists of three sub-steps. Firstly, input data for the calculations are prepared by the Support Functions Library. This procedure is based on the operating mode, either the in-memory data are used or the data are read from the file(s). Secondly, the actual calculations are performed. Finally, the outputs of the calculations are processed based on the run mode. In case of the single-step mode, they are stored to files, while in pipeline-processing they are passed to the next step and optionally saved to files.

This architecture defines a clear interface between calculation functions and data preparation / handling. It uses clearly defined data structures, and makes the future additions to the pipeline easier to implement.

The individual steps of the processing pipeline are described in detail in subsequent sections of this SDD. There is a section for each library; each section describes the purpose, requirements and design considerations that have been considered for that particular library.

3.1.4.2. Initialization

The following steps are performed during the initialization of the *autoSaint* software.

3.1.4.2.1. Logging

The software logs a start-up message.

3.1.4.2.2. Verifying the User

The software verifies the operating system user name. If the current user name is “root” (super user), the software generates an error and terminates.

3.1.4.2.3. Configuration

The software parses the parameters provided in the command line. The following command line parameters are mandatory:

- Sample ID
- Database connect string, the file containing the connect string or a parameter specifying that the default file containing the connect string shall be used. If the connection fails, the software generates an error and terminates.

Afterwards, the software connects to the database and reads the parameters provided in the **GARDS_SAINT_DEFAULT_PARAMS** database table. For a list of parameters, see the Appendix II, Configuration Parameters.

For any unspecified parameters, the default values will be used, if defined.

SQL query used when reading processing parameters:

```
SELECT NAME, VALUE, MODDATE FROM GARDS_SAINT_DEFAULT_PARAMS
```

After parsing the configuration, the software verifies correctness of configuration and exits if there is a problem in the configuration (e.g. missing mandatory parameters). The parameter rules are defined in the Appendix II.

The software then stores the actual configuration parameters to the **GARDS_SAINT_PROCESS_PARAMS** database table, using SAMPLE_ID as a primary key.

SQL queries used when storing used processing parameters:

```
DELETE GARDS_SAINT_PROCESS_PARAMS WHERE SAMPLE_ID=%d

INSERT INTO GARDS_SAINT_PROCESS_PARAMS (SAMPLE_ID, NAME, VALUE) VALUES (%sampleId,
'%parameterName', '%parameterValue')

INSERT INTO GARDS_SAINT_PROCESS_PARAMS (SAMPLE_ID, NAME, VALUE) VALUES (%sampleId,
'%parameterName', NULL)
```

If the help function was requested in the input parameters, the software displays the description of available parameters and exits.

If the version string was requested in the input parameters, the software displays the version string and exits.

3.1.4.2.4. Check whether the Sample was Already Processed

The software checks whether the sample was already processed to avoid reprocessing of an already processed sample. This check is based on the STATUS attribute in **GARDS_SAMPLE_STATUS** table.

- If the STATUS is ‘U’ (unprocessed) or ‘A’ (currently under processing / failed processing), the sample is processed.
- If the STATUS is ‘P’ (processed), and the overwrite flag is not set, the sample is not reprocessed
- If the STATUS is ‘P’ and the overwrite flag is set, the sample is reprocessed and a warning message is written to the log file. The previous output in the database and the file system is overwritten.
- If the STATUS has any other value than ‘U’,’A’ or ‘P’, an error message is written to the log file and the sample is not processed.

At the beginning of the processing, the STATUS is set to ‘A’. If the processing is successful, the STATUS is set to ‘P’, if the processing fails the STATUS remains set to ‘A’.

SQL queries used when reading and writing processing status:

```
SELECT STATUS FROM GARDS_SAMPLE_STATUS WHERE SAMPLE_ID = %sampleId

UPDATE GARDS_SAMPLE_STATUS SET STATUS = '%newSampleStatus' WHERE SAMPLE_ID = %sampleId

DELETE FROM GARDS_COMMENTS WHERE (SAMPLE_ID = %sampleId) AND (UPPER(ANALYST) NOT LIKE
'%%INPUT%%' OR  (ANALYST IS NULL))
```

3.1.4.2.5. Preparing the Data for Processing

The software prepares the data needed for the processing of the sample. The following steps are performed:

- Read sample data from the SQL database
- Read sample spectrum data
- For Xenon noble gas processing, read preliminary samples data
- Get MRP coefficients
- Prepare energy and resolution arrays for the calibration. See section 3.4.4.1 for details.

SQL queries used to read sample data:

```
SELECT GSD.SITE_DET_CODE, GSD.SAMPLE_ID, GSD.STATION_ID, GSD.DETECTOR_ID,
GSD.INPUT_FILE_NAME, GSD.SAMPLE_TYPE, GSD.DATA_TYPE, GSD.GEOMETRY, GSD.SPECTRAL_QUALIFIER,
GSD.TRANSMIT_DTG, GSD.COLLECT_START, GSD.COLLECT_STOP, GSD.ACQUISITION_START,
GSD.ACQUISITION_STOP, GSD.ACQUISITION_REAL_SEC, GSD.ACQUISITION_LIVE_SEC, GSD.QUANTITY,
GSD.MODDATE , CAST(NVL(GSD.ACQUISITION_REAL_SEC, 0.0) AS NUMBER), GS.STATION_CODE FROM
GARDS_SAMPLE_DATA GSD, GARDS_STATIONS GS WHERE GSD.SAMPLE_ID = %sampleId AND GSD.STATION_ID
= GS.STATION_ID

SELECT XE_VOLUME FROM GARDS_SAMPLE_AUX WHERE SAMPLE_ID=%sampleId

SELECT DIR, DFILE, CAST(FOFF AS NUMBER(11,1)), CAST(DSIZE AS NUMBER(11,1)) FROM FILEPRODUCT
WHERE TYPEID = %fileproductTypeId AND CHAN = '%sampleId'

SELECT CHANNELS, CAST(NVL(START_CHANNEL,-1) AS NUMBER) FROM GARDS_SPECTRUM WHERE
SAMPLE_ID=%sampleId
```

3.1.4.3. Calibration

During the calibration, various processing parameters are recalculated. The calibration consists of the following steps:

- Calculate Baseline of the main sample spectrum (and of preliminary spectra for Xenon samples)
- Calculate LC of the main sample spectrum (and of preliminary spectra for Xenon samples)
- Calculate SCAC of the main sample spectrum (and of preliminary spectra for Xenon samples)
- Perform the initial peak search for energy calibration
- Identify reference peaks for energy calibration
- Perform energy calibration and competition
- Perform the initial peak search for resolution calibration (with variable fwhm)
- Identify reference peaks for resolution calibration
- Perform resolution calibration and competition

The details of the individual steps are described in the section 3.2.

3.1.4.4. Pipeline Processing

The sample is analyzed using the processing parameters that won in the competition performed during energy and resolution calibration.

The following steps are performed:

- Calculate Baseline of the main sample spectrum (and of preliminary spectra for Xenon samples)
- Calculate LC of the main sample spectrum (and of preliminary spectra for Xenon samples)
- Calculate SCAC of the main sample spectrum (and of preliminary spectra for Xenon samples)
- Perform peak search
- For particulate samples:
    - Identify nuclides
- For Xenon samples:
    - Perform Xenon analysis
- Calculate activities and MDCs
- Perform categorization
  Note: This step is optional and it is currently not used in the IDC
- Perform QC checks
- Set processing status

The details of individual steps are described in sections 3.2 and 3.2.4.6.

3.2. Scientific Calculations Library

3.2.1. Overview

The Scientific Calculations Library contains all scientific functionality of the software, such as baseline, SCAC and LC calculations and nuclide identification. These functions are used by the Pipeline Wrapper to execute the processing steps.

3.2.2. Dependencies

The Scientific Calculations Library depends on the Infrastructure Library to perform logging and to access the configuration. The interfaces to the library are defined by their respective header files. There is one interface function for each high-level function defined in section 3.2.4.

The input data are prepared and the outputs parsed by the Supporting Functions Library and the Data Access Library functions.

3.2.3. Requirements

There are no explicit requirements listed in [AUTO_SAINT_SRS] and [AUTO_XE_SAINT_SRS]. The design of the scientific calculations is based on the existing source code, the prototypes and the results of discussions with IDC staff.

Note: Additional requirements were identified and corresponding software changes designed and implemented in the test use of autoSaint.

3.2.4. Design decisions

The following high-level functions are defined in the Scientific Calculations Library:

- Calculate baseline
- Calculate SCAC
- Calculate LC
- Find peaks
- Nuclide identification
- Xenon Analysis

3.2.4.1. Calculate Baseline

The goal of this function is to compute the level of noise across the whole spectrum. It is based on the "lawn mower" algorithm, which cuts out each peak identified in a given energy range with respect to the slope of the selected spectrum area.

Before and after applying the "lawn mower" algorithm, the selected part of the spectrum is smoothed.

The number of times the "lawn mower" algorithm is applied depends on the part of the spectrum that is considered.

The energy boundaries and the number of passes are configurable for each detector ID and data type and it is defined in the **GARDS_BASELINE** database table. The following Table 2 shows the working example number of passes of the "lawn mower" algorithm used for each part of the spectrum. Different configurations can be defined for different types of spectra.

Table 2- Number of passes of “lawn mover”

NR of passes	Energy minimum	Energy maximum	Other condition
4	-	55	spectrum > 1
2	63	65	-
5	62	70	-
15	67	79	-

NR of passes	Energy minimum	Energy maximum	Other condition
15	67	96	-
2	95	120	-
2	117	138	-
4	130	160	-
4	504	516	-
4	2355	2390	-
10		155	-

Description of "Lawn mower" algorithm

For each channel j we define a channel interval j – δ1, j + δ2 where δ1 and δ2 are the equivalent in channels of 2*FWHM(j). If the channel j happens to be the one with maximum counts in this interval, it is a good candidate to be the centroid of a potential peak. In this case the original spectrum in the selected interval is replaced with a straight line from j – δ1 to j + δ2.

3.2.4.2. Calculate SCAC

The Single Channel Analyzer Curve (SCAC) is the spectrum as "seen by a sliding single channel analyzer". It is computed by a smoothing of the spectrum.

$$SCAC(i) = \frac{\delta 2 * Spectrum(i - \delta 1 - 1) + \delta 2 * Spectrum(i + \delta 1 + 1) \sum_{i-\delta 1}^{i+\delta 1} Spectrum(i)}{1.25 * Fwhmc(i)}$$

$$\delta 1 = E((((1.25 * Fwhmc(i)) - 1)/2) + 0.000001)$$

$$\delta 2 = (((1.25 * Fwhmc(i)) - 1)/2) + 0.000001 - \delta 1$$

3.2.4.3. Calculate LC

The goal of this function is to compute the "critical level" named LC.

LC is equal to the Baseline plus the uncertainty of the Baseline considering a given risk level.

So the regions where SCAC is above LC are most likely due to actual peaks and not to random noise.

The formula is:

$$\forall i = 1..m : LC_i = B_i + k.\sqrt{\frac{B_i}{1.25R_i}} \quad ,$$

where

$m = number\ of\ \ channels\ in\ the\ spectrum$
$LC_i = channel\ i\ of\ \ LC$
$B_i = channel\ i\ of\ \ Baseline$
$R_i = resolution\ of\ \ peaks\ around\ \ channel\ i$
$k = risk\ level\ in\ this\ region\ of\ \ the\ spectrum$

3.2.4.4. Find Peaks

The aim of this calculation is to find all the peaks in the spectrum.
The peaks to be identified have to satisfy the simultaneous equations:

$$\forall j = 1..m : S_j - B_j = \sum_{i=0..n} \frac{w_i.A_i}{\sigma_i.\sqrt{2.\pi}} * e^{\frac{(e_j - c_i)^2}{2.\sigma_i^2}} \quad ,$$

where

$m = number\ of\ \ channels\ in\ the\ spectrum$
$n = number\ of\ \ peaks\ in\ the\ spectrum$
$S_j = channel\ \ j\ of\ \ the\ spectrum$
$B_j = channel\ \ j\ of\ \ the\ baseline$
$c_i = centroid\ of\ peak\ i$
$\sigma_i = sigma\ of\ \ peak\ i$
$wi = channel\ width\ around\ \ peak\ i\ centroid$
$A_i = area\ of\ \ peak\ i$
$e_j = energy\ of\ \ channel\ \ j$

σ, e and w are calculated with the data of calibration arrays.
Initially, peaks are searched one by one from left to right with a left Gaussian fitting.
Then areas and centroids are tuned simultaneously and iteratively with a least square fitting.
Peaks whose magnitude is less than noise are discarded.

For each found peak, the following additional values are calculated:

Area error

This value is computed based on the energy of the peak centroid i.

$$AreaError = \sqrt{\frac{(SCAC_i + B_i * 1.25 * Résolution_i}{0.85891}}$$

Efficiency

This value is computed based on the energy of the found peak. The coefficients are given as input data.

$$c = \log(Coeff\,0 / ene)$$

$$Efficiency = Exp(\sum_{i=1..n} Coeff_i * c^{i-1})$$

ene : centroid energy of the current peak

Detectability (ene is the centroid value of the considered peak)

$$\forall i = ene - 2...ene + 2 : Detectability = Max(\frac{SCAC_i - B_i}{LC_i - B_i})$$

Finally, the filter is applied on the peaks to filter out erroneous unphysical peaks:

- Peaks with negative detectability are discarded.
- If the peak detectability is positive but less than one, the peak area condition is applied. If a peak area is greater than
  $1.25 \times fwhm[centroid] \times \max_{peak\ channels}(LC - baseline) / 0.8591$,
  the peak is discarded because of unphysical area.

3.2.4.5. Nuclides Identification

The goal of this function is to identify the radionuclide responsible for the peaks found in the previous step. For each peak, a list of radionuclide with the associated contribution (in %) is given.

The routine structure is the following:

1. Find nuclides where the key line energy is close (tolerance to be set as input value) to one of the peaks found.
2. Reject nuclides where a support line (with higher detectability than the key line in the current spectrum) does not justify them.
3. Reject nuclides after interference checking (in other words, reject nuclides whose key-line is actually a line of another nuclide present)
4. Identify the other lines belonging to the selected nuclide (also using interference check).

5. Mark the unidentified peaks as unknown.

This routine uses the IDC Nuclide Library and will correct for true coincidence where an Isotope Response Function (IRF) is available.

Parameters for energy tolerance and error tolerance are given as input.

3.2.4.6. Xenon Analysis

The goal is to compute the area of each gamma peak associated to Xe isotopes.

Two different methods are used (called Method 1 and Method 2) and both method algorithms are explained in the following sections.

There are four Xe isotopes

- Two metastable isotopes Xe131m and Xe133m
- Two non-metastable isotopes Xe133 end Xe135

Each isotope is associated with one gamma peak and four peaks in the X-ray region (also simply referred to as “X” below).

Knowing the energy and the probability of each X peak it is possible to determine the shape of the X spectrum:

$$Gausx(chan) = somme_{pic} (Laurantian(chan, energy_{pic}) * probability_{pic})$$

where

Gausx	: X spectrum shape of the considered isotope
chan	: channel
$energy_{pic}$	: peak energy
$probability_{pic}$	: peak probability

Once obtained this shape is normalized so that the area is equal to one:

Func(chan) = Gausx (chan) sum Gausx

where

Func	: normalized multiplet
chan	: channel
sum Gausx	: summation of all the Gausx channels

Comments:

- The two metastable isotopes share the same normalized multiplet
- The two non metastable isotopes share another normalized multiplet

For each Xe isotope, the ratio between the multiplet area and the gamma peak area can be computed:

Ratio = multiplet area / gamma peak area

$= \text{sum}_{k=1..4} (\text{effx}_k * \text{branchx}_k) / \text{eff}_g * \text{branch}_g$

where

Ratio	: X and Gamma area ratio
effx_k	: efficiency of the X peak number k
branchx_k	: branch value the X peak number k
eff_g	: gamma efficiency
branch_g	: gamma branch value

3.2.4.6.1. Method 1 algorithm

The goal of this method is to approximate the full spectrum based on the gamma peak area. There is one equation for each channel in the X ray region:

$\text{sum}_{i=1..4} (X_i * \text{Func}_i(\text{chan}) / \text{Ratio}_i) = \text{Spectrum}(\text{chan}) - \text{Baseline}(\text{chan}) + \text{err}(\text{chan})$ (1)

where

i	: Xe isotope index
X_i	: unknown value of the area of the gamma peak for the isotope number i
Func_i	: normalized multiplet of the isotope number i
Ratio_i	: ratio of the X and Gamma area of the isotope number i
Spectrum	: measured spectrum
Baseline	: spectrum baseline
Err	: spectrum error

For each isotope, the gamma area can be defined as follows:

$X_i = A_i + \text{err}(i)$ (2)

where

X_i	: unknown value of the area of the gamma peak for the isotope number i
A_i	: gamma peak area of the isotope number i measured in the spectrum
err(i)	: error on the estimated value of A_i

Knowing that the gamma peaks are very low, the area is estimated based on the SCAC value around the theoretical value of the energy of each peak. Furthermore, the criteria SCAC > LC must not be taken into account.

$$A_i = \text{Max}\ (SCAC(chan) - Baseline(chan)) * 1.25*fwhmc(chan)/0.8591$$

where

A_i	: gamma peak area of the isotope number i measured in the spectrum
SCAC	: computed SCAC
Baseline	: computed baseline
fwhmc	: fwhm in channels

Using equations (1) and (2), an equation system can be created and solved with the least mean square method. The measurement uncertainties are also taken into account.

The results are the gamma area of each Xe isotope, and the associated uncertainties are obtained by the root square value of the terms found in the main diagonal of the covariance matrix.

Before storing the result, the covariance matrix values are adjusted by the decrease factor of the corresponding isotope:

$$cov_{adj}[i,isotope] = cov[i,isotope] * fAct*fAct$$

where

```
fAct = CCF / ( abundance[isotope] * detectorEfficiency *
acquisitionLifeTime ) * 1e03
CCF is a Coincidence Correction Factor
```

3.2.4.6.2. Method 2 algorithm

This method is based on the algorithm used in the method 1. In addition, decrease in peak area for each isotope due to its decay is also used in the calculation. The algorithm makes use of preliminary spectra.

The full spectrum is measured between t_0 and t_{tot}.

With a preliminary spectrum given at the time t, the area of a given isotope i can be computed as follow:

$$A_{it} = A_{itot} * fact_{it}$$

where

A_{it}	: gamma peak area of the given isotope measured between t_0 and t
A_{itot}	: gamma peak area of the given isotope measured between t_0 and t_{tot}

$fact_{it}$: decrease factor of the isotope i at t time

The decrease factor is computed as follow:

$fact_{it} = [1-\exp(-lambda_i * (t - t_0))] / [1-\exp(-lambda_i * (t_{tot} - t_0))]$

where

$fact_{it}$: decrease factor of the isotope i at t

$lambda_i$: i decrease coefficient value associated to the concidered isotope

For each given preliminary spectrum and for the full spectrum the equations (1) and (2) are reused but X_i is replaced by $X_i * fact_{it.}$

The resolution of the equation system is done in the same way as for method 1.

3.2.4.6.3. Laurantian

A Laurantian is the mean of Gaussians computed around the energy of a given peak.

The Laurantian is parameterized with a parameter named gamma. When gamma is set to 0 the Laurentian is reduced to a pure Gaussian.

In the algorithms used for Xe isotope determination, the gamma parameter value is set to 15.518870.

The Laurentian is computed as follow:

Laurantian (ener) = 1/99 * Sum $_{i=1..99}$ (gaussian(ener, mu(i), sigma))

where

ener : energy of the computed channel

mu(i) : Gassian centroid for the i indices

sigma : Gaussian sigma value

Gaussians are distributed around mu0 according to the i indices by using the following law :

mu(i) = mu0 + (gamma/500)*tan((i/100-0.5)*pi)

3.3. Additional Calculations Library

3.3.1. Overview

The Additional Calculation Library contains additional radionuclide calculation functions needed to perform the processing pipeline, like a categorization and a QC check. These functions are used by a Pipeline Wrapper to execute the processing steps.

3.3.2. Dependencies

The Additional Calculations Library depends on the Infrastructure Library to perform logging and to access the configuration. The interfaces to the library are defined by their respective header files.

The input data are prepared and the outputs parsed by the Supporting Functions Library and the Data Access Library functions.

3.3.3. Requirements

There are no explicit requirements listed in [AUTO_SAINT_SRS] and [AUTO_XE_SAINT_SRS]. The design of the additional calculations is based on the existing source code and prototypes and on the results of discussions with the IDC.

Note: Additional requirements were identified and corresponding software changes designed and implemented in the test use of autoSaint.

3.3.4. Design decisions

The following high-level functions are defined in the Additional Calculations Library:

- Categorization
- QC
- Calculation of calibration arrays
- Recalculation of processing parameters (calibration)
- Competition
- Activities Calculation
- Identification of reference peaks

3.3.4.1.Identification of Reference Peaks

The goal of this function is to find particular, well-known peaks in the spectrum.

The results of the peak search with variable sigma parameter are compared to the list of well known reference peaks defined in the database table **GARDS_REF_LINES** (or **GARDS_XE_REF_LINES** for Xenon samples).

The following steps are performed:

- Filter out the peaks with an area smaller than the area threshold
- Associate found peaks to the reference lines by searching for the closest peak for each reference line. The peak is linked to the reference line only if the distance between reference line and peak energy is smaller than a configurable threshold.

3.3.4.2. Categorization

Note: The categorization by autoSaint can be performed for particulate samples only and it is currently not in use in the IDC.

The categorization assigns one of the category levels 1 to 5 to each sample.

First, individual nuclides are categorized:

- Nuclide template is loaded of it exists
- The relevance of the nuclide is determined
- The nuclide type is identified to determine whether the nuclide is natural or cosmic
- Natural non-relevant nuclides are assigned the category level 2
- For non-natural and/or relevant nuclides, if the template exists, it is used to determine the category level.
- If the template does not exist, the category level of a nuclide is defined based on relevance, nuclide count in the last month, nuclide count in history and on other attributes.

Then, the category level of the sample is determined based on the nuclide categorization results. The category of the sample will be set equal to the highest category of its nuclides, with special treatment applied to the category level 4 nuclides.

SQL queries used in categorization:

```
SELECT NID_FLAG, NAME, ACTIV_KEY, TYPE FROM GARDS_NUCL_IDED WHERE SAMPLE_ID = %sampleId

SELECT STATION_ID, DETECTOR_ID, NAME, METHOD_TYPE, CAST(NVL(UPPER_BOUND,0.0) AS NUMBER),
CAST(NVL(LOWER_BOUND,0.0) AS NUMBER), CAST(NVL(CENTRAL_VALUE,0.0) AS NUMBER),
TO_CHAR(BEGIN_DATE,'YYYY-MM-DD HH24:MI:SS'), CAST(NVL(ABSCISSA,0.0) AS NUMBER) FROM
GARDS_CAT_TEMPLATE WHERE STATION_ID = %stationId AND NAME = '%nuclideName' AND BEGIN_DATE <
to_date('%acquisitionStart', 'YYYY/MM/DD HH24:MI:SS') AND END_DATE >
to_date('%acquisitionStart', 'YYYY/MM/DD HH24:MI:SS')

SELECT STATION_ID, DETECTOR_ID, NAME, METHOD_TYPE, CAST(NVL(UPPER_BOUND,0.0) AS NUMBER),
CAST(NVL(LOWER_BOUND,0.0) AS NUMBER), CAST(NVL(CENTRAL_VALUE,0.0) AS NUMBER),
TO_CHAR(BEGIN_DATE,'YYYY-MM-DD HH24:MI:SS'), CAST(NVL(ABSCISSA,0.0) AS NUMBER) FROM
GARDS_CAT_TEMPLATE WHERE STATION_ID = %stationId AND NAME = '%nuclideName' AND BEGIN_DATE <
to_date('%acquisitionStart', 'YYYY/MM/DD HH24:MI:SS') AND END_DATE IS NULL

SELECT TYPE FROM GARDS_RELEVANT_NUCLIDES WHERE NAME = '%nuclideName' AND SAMPLE_TYPE =
'%sampleType'

SELECT TYPE FROM GARDS_NUCL_LIB WHERE NAME = '%nuclideName'

SELECT CAST(COUNT(GSC.ACTIVITY) AS NUMBER(11,1)) FROM GARDS_SAMPLE_CAT GSC,
GARDS_SAMPLE_DATA GSD, GARDS_READ_SAMPLE_STATUS GSS WHERE GSC.SAMPLE_ID = GSD.SAMPLE_ID AND
GSC.SAMPLE_ID = GSS.SAMPLE_ID AND GSD.STATION_ID = %stationId AND GSC.NAME = '%nuclideName'
AND GSS.STATUS IN ('R' ,'Q') AND GSS.CATEGORY IS NOT NULL AND GSD.COLLECT_STOP BETWEEN
to_date('%collectStop', 'YYYY/MM/DD HH24:MI:SS')-30 AND to_date('%collectStop', 'YYYY/MM/DD
HH24:MI:SS')

SELECT GSC.ACTIVITY FROM GARDS_SAMPLE_CAT GSC, GARDS_SAMPLE_DATA GSD,
GARDS_READ_SAMPLE_STATUS GSS WHERE GSC.SAMPLE_ID = GSD.SAMPLE_ID AND GSC.SAMPLE_ID =
GSS.SAMPLE_ID AND GSD.STATION_ID = %stationId AND GSC.NAME = '%nuclideName' AND GSS.STATUS
in ('R' ,'Q') AND GSS.CATEGORY IS NOT NULL AND HOLD = 0 AND GSD.COLLECT_STOP <
to_date('%collectStop', 'YYYY/MM/DD HH24:MI:SS') ORDER BY GSD.COLLECT_STOP DESC

SELECT GSC.ACTIVITY FROM GARDS_SAMPLE_CAT GSC, GARDS_SAMPLE_DATA GSD,
GARDS_READ_SAMPLE_STATUS GSS WHERE GSC.SAMPLE_ID = GSD.SAMPLE_ID AND GSC.SAMPLE_ID =
GSS.SAMPLE_ID AND GSD.STATION_ID = %stationId AND GSC.NAME = '%nuclideName' AND GSS.STATUS
IN ('R' ,'Q') AND GSS.CATEGORY IS NOT NULL AND HOLD = 0 AND GSD.COLLECT_STOP <
to_date('%collectStop', 'YYYY/MM/DD HH24:MI:SS') ORDER BY GSD.COLLECT_STOP DESC

SELECT GSD.SAMPLE_ID, to_char(GSD.ACQUISITION_START, 'YYYY-MM-DD HH24:MI:SS') FROM
GARDS_SAMPLE_DATA GSD, GARDS_READ_SAMPLE_STATUS GSS, GARDS_SAMPLE_CAT GSC WHERE
HSD.STATION_ID = %stationId AND GSD.DETECTOR_ID = %detectorId AND GSD.ACQUISITION_START <
to_date ('%acquisitionStart','YYYY/MM/DD HH24:MI:SS') AND GSD.SAMPLE_ID = GSS.SAMPLE_ID AND
GSD.SAMPLE_ID = GSC.SAMPLE_ID AND GSS.STATUS IN ( 'R', 'Q' ) AND GSS.CATEGORY IS NOT NULL
AND GSC.NAME = '%nuclideName' AND GSC.HOLD = 0 ORDER BY ACQUISITION_START DESC

SELECT CAST(NVL(UPPER_BOUND,0.0) AS NUMBER), CAST(NVL(LOWER_BOUND,0.0) AS NUMBER),
CAST(NVL(CENTRAL_VALUE,0.0) AS NUMBER) FROM GARDS_SAMPLE_CAT WHERE SAMPLE_ID = %sampleId
AND NAME = '%nuclideName'

UPDATE GARDS_SAMPLE_STATUS SET AUTO_CATEGORY = %newAutoCategory, CATEGORY = NULL WHERE
SAMPLE_ID = %sampleId

SELECT AUTO_CATEGORY FROM GARDS_SAMPLE_STATUTS WHERE SAMPLE_ID = %sampleId
```

3.3.4.3. QC

The quality control (QC) consists of several independent QC checks performed at the end of sample processing. The QC checks performed are shown in Table 3. Each QC check can be separately enabled or disabled by the *autoSaint* configuration.

Table 3 – List of QC checks

Name	Condition
Acquisition time	The acquisition time of a sample must be longer than or equal to 20 hours.
Collection time	The collection time of a sample must be

Name	Condition
	between 21.6 and 26.4 hours
Decay time	The decay time of a sample must be between 21.6 and 26.4 hours
Reporting time	The reporting time of a sample must be shorter than 72 hours.
Air volume	The total air volume must be at least 500 cubic meters.
Collection gaps	The collection gaps must be in the range 0-30 minutes.
Preliminary samples	The number of preliminary samples must correspond to the sample acquisition time.
Flags	Be-7 FWHM test: FWHM (477.5) < 1.7 Ba-140 MDC test: MDC within defined limits
Flow 500	SOH flow rate higher than or equal to defined threshold.
Flow GAP	No gaps in flow data.
Flow ZERO	SOH blower data should be complete.
Flow	Measured quantity should match calculated quantity.
Drift MRP	The peak attributes are checked for drift problems within 10 days.
Drift 10 days	The peak attributes are checked for drift problems within 10 to 30 days.
Categorization	Auto category present and with value less than 4.
ECR	Check whether MRP_A or MRP_M was used for both ECR and RER calibration.
Nuclide identification	At least 80% of the peaks are identified.

SQL queries used in QC checks:

```
SELECT MDA FROM GARDS_NUCL_IDED WHERE (SAMPLE_ID = %sampleId) AND (NAME LIKE 'BA-140')

SELECT MDA_MIN, MDA_MAX FROM GARDS_MDAS2REPORT WHERE (NAME = 'BA-140') AND (SAMPLE_TYPE =
'%sampleType') AND (DTG_BEGIN < to_date ('%acquisitionStart','YYYY/MM/DD HH24:MI:SS')) AND
(DTG_END IS NULL OR DTG_END > to_date ('%acquisitionStart','YYYY/MM/DD HH24:MI:SS'))

SELECT D1.SAMPLE_ID, D1.COLLECT_STOP, D2.COLLECT_START FROM GARDS_SAMPLE_DATA D1,
GARDS_SAMPLE_DATA D2 WHERE D1.DATA_TYPE = D2.DATA_TYPE AND D1.SPECTRAL_QUALIFIER =
D2.SPECTRAL_QUALIFIER AND D1.STATION_ID = D2.STATION_ID AND D1.DETECTOR_ID = D2.DETECTOR_ID
AND (D2.ACQUISITION_STOP - D1.ACQUISITION_STOP) < 5.0 AND D1.ACQUISITION_STOP <
D2.ACQUISITION_STOP AND D2.SAMPLE_ID = (%sampleId) ORDER BY D1.ACQUISITION_STOP DESC

SELECT D1.SAMPLE_ID, D1.ACQUISITION_START, D1.ACQUISITION_STOP FROM GARDS_SAMPLE_DATA D1,
GARDS_SAMPLE_DATA D2 WHERE D1.COLLECT_START = D2.COLLECT_START AND D1.COLLECT_STOP =
D2.COLLECT_STOP AND D1.ACQUISITION_START = D2.ACQUISITION_START AND D1.SPECTRAL_QUALIFIER =
'PREL' AND D2.SAMPLE_ID = (%sampleId) ORDER BY D1.ACQUISITION_STOP

SELECT F.NAME FROM GARDS_SAMPLE_FLAGS S, GARDS_FLAGS F WHERE S.FLAG_ID = F.FLAG_ID AND
S.RESULT = 0 AND S.SAMPLE_ID = (%sampleId)

SELECT * FROM GARDS_PEAKS WHERE ENERGY >= 1460.3 AND ENERGY <= 1461.3 AND SAMPLE_ID =
(%sampleId)

SELECT VALUE, DTG_BEGIN, DTG_END FROM GARDS_SOH_NUM_DATA WHERE STATION_ID = %stationId AND
PARAM_CODE = %paramCode AND GARDS_SOH_NUM_DATA.DTG_BEGIN <
to_date('%collectStop','YYYY/MM/DD HH24:MI:SS') AND GARDS_SOH_NUM_DATA.DTG_END >
to_date('%collectStart','YYYY/MM/DD HH24:MI:SS') AND (GARDS_SOH_NUM_DATA.DTG_END -
to_date('%collectStop','YYYY/MM/DD HH24:MI:SS')) < (1/24) ORDER BY DTG_BEGIN, DTG_END

SELECT VALUE, DTG_BEGIN, DTG_END FROM GARDS_SOH_CHAR_DATA WHERE STATION_ID = %stationId AND
PARAM_CODE = %paramCode AND GARDS_SOH_CHAR_DATA.DTG_BEGIN <
to_date('%collectStop','YYYY/MM/DD HH24:MI:SS') AND GARDS_SOH_CHAR_DATA.DTG_END >
to_date('%collectStart','YYYY/MM/DD HH24:MI:SS') AND (GARDS_SOH_CHAR_DATA.DTG_END -
to_date('%collectStop','YYYY/MM/DD HH24:MI:SS')) < (1/24) ORDER BY DTG_BEGIN, DTG_END

SELECT SAMPLE_REF_ID FROM GARDS_SAMPLE_AUX WHERE SAMPLE_ID = %sampleId

SELECT SAMPLE_ID, ACQUISITION_START, ACQUISITION_STOP FROM GARDS_SAMPLE_DATA WHERE
DATA_TYPE = 'Q' AND STATION_ID = %stationId AND DETECTOR_ID = %detectorId AND ( to_date(
'%acquisitionStart','YYYY/MM/DD HH24:Mi:SS') - ACQUISITION_STOP ) < %gapThreshold

SELECT SAMPLE_ID, ACQUISITION_STOP FROM GARDS_SAMPLE_DATA WHERE DATA_TYPE = 'Q' AND
STATION_ID = %stationId AND DETECTOR_ID = %detectorId AND (to_date('%acquisitionStop',
'YYYY/MM/DD HH24:Mi:SS') - ACQUISITION_STOP) < %d AND SAMPLE_ID < %sampleId ORDER BY
ACQUISITION_STOP DESC

SELECT SAMPLE_ID, ACQUISITION_STOP FROM GARDS_SAMPLE_DATA WHERE DATA_TYPE = 'Q' AND
STATION_ID = %stationId AND DETECTOR_ID = %detectorId AND (to_date('%acquisitionStop',
'YYYY/MM/DD HH24:Mi:SS') - ACQUISITION_STOP) < %highThreshold AND
(to_date('%acquisitionStop', 'YYYY/MM/DD HH24:Mi:SS') - ACQUISITION_STOP) > %lowThreshold
AND SAMPLE_ID < %sampleId ORDER BY ACQUISITION_STOP DESC

SELECT SAMPLE_ID, CENTROID, FWHM, AREA FROM GARDS_PEAKS WHERE SAMPLE_ID IN (%sampleId1,
%sampleId2) AND AREA > %threshold ORDER BY SAMPLE_ID, CENTROID

SELECT PEAK_ID FROM GARDS_PEAKS WHERE (SAMPLE_ID = %sampleId) AND (IDED=1)

SELECT PEAK_ID FROM GARDS_PEAKS WHERE (SAMPLE_ID = %sampleId) AND (IDED!=1)

DELETE FROM GARDS_QC_RESULTS WHERE SAMPLE_ID = %sampleId

INSERT INTO GARDS_QC_RESULTS (SAMPLE_ID, TEST_NAME, FLAG, QC_COMMENT ) VALUES ( %sampleId,
'%testName', '%testResult', '%comment' )
```

3.3.4.4. Calibration Arrays

The energy and resolution calibration arrays are used during the calibration part of sample processing.

In case that the MRP_A calibration coefficients are used to calculate the arrays, the polynomial coefficients are read from the database.

In case of INPUT, the polynomial parameters is calculated using the least square polynomial fitting of the input pairs defined in the input sample.

Individual calibration coefficients sets used in *autoSaint* are described in section 3.4.4.1.

Once the polynomial parameters are available, the energy and resolution arrays are calculated.

There are two options to calculate the energy calibration array. The option used can be specified in the configuration of *autoSaint* software. The formulas associated with these options are based on Saint Matlab prototypes.

The first (default) option uses the formula:

$$\forall i = 1..m : E_i = \sum_{j=0..n} c_j . x_i^j \quad ,$$

where

$m = number\ of\ \ channels\ in\ the\ spectrum$

$E = energy$

$n = energy\ regression\ \ polynom\ degree$

$c_j = energy\ regression\ \ polynom\ coefficients$

$x = [1, 2, .., m]$

The second option uses the formula

$$\forall i = 1..m : E_i = \frac{\left(\sum_{j=0..n} c_j . x_i^j + \sum_{j=0..n} c_j . y_i^j \right)}{2} \quad .$$

where

$m = number\ of\ \ channels\ in\ the\ spectrum$

$E = energy$

$n = energy\ regression\ \ polynom\ degree$

$c_j = energy\ regression\ \ polynom\ coefficients$

$x = [1, 2, .., m]$

$y = [0,1,..., m-1]$

The resolution array can be calculated using the formula

$$\forall i = 1..m : R_i = \sqrt{\sum_{j=0..n} c_j . E_i^j} \quad ,$$

where

$m = number\ of\ \ channels\ in\ the\ spectrum$

$E = energy$

$R = resolution$

$n = resolution\ regression\ \ polynom\ degree$

$c_j = resolution\ regression\ \ polynom\ coefficients$

$x = [1, 2, .., m]$

3.3.4.5. Recalculate Processing Parameters

After the reference peak search, the INITIAL energy and resolution regression coefficients are calculated based on the centroid's channel, energy and resolution calculated for each reference peak.

A least-square fit using the Singular Value Decomposition method is used to fit the polynomial function to the pairs defined by reference peaks.

As a first step, the number of reference peaks found in the initial peak search is compared to a configurable threshold. If not enough peaks are found, the INITIAL coefficients are not calculated.

3.3.4.5.1. Energy Calibration

The energy coefficients calibration is performed using a least-square fitting of pairs $(channel_i, energy_i), i = 1..no\ of\ \ reference\ \ peaks$ by the polynomial function $ecr(channel) = \sum_{i=0}^{M} a_i \cdot channel^i$. The error is defined by vector

$$\Delta energy = \begin{pmatrix} \Delta channel_1 \cdot \dfrac{energy_1}{channel_1} \\ \Delta channel_2 \cdot \dfrac{energy_2}{channel_2} \\ \cdots \\ \Delta channel_N \cdot \dfrac{energy_N}{channel_N} \end{pmatrix} .$$

In the calculation, $channel_i$ is the centroid channel as calculated by the initial peak search and $energy_i$ is the reference line energy. M is a degree of the fitted polynomial function and is configurable in the software, and N is the number of reference peaks.

The outputs of the least-square fitting are the vector of coefficients of the fitted polynomial function $A = \begin{pmatrix} a_0 \\ a_1 \\ ... \\ a_M \end{pmatrix}$ and the covariance matrix $\mathrm{cov}(A)$.

3.3.4.5.2. Resolution Calibration

The resolution and resolution error values provided by the initial peak search are in the form of resolution in channel. They must be first recalculated to resolution in energy. The recalculation is done using the energy coefficients A calculated in the energy calibration:

$$resolution_{keV} = resolution_{channel} \cdot \frac{d(energy)}{d(channel)} =$$
$$= resolution_{channel} \cdot \left(a_1 + 2 \cdot a_2 \cdot channel + ... + M \cdot a_M \cdot channel^{M-1}\right)$$

where $channel$ is the corresponding centroid channel calculated in the initial peak search.

The same recalculation applies to resolution error.

As the fitting function for resolution is $rer(channel) = \sqrt{\sum_{i=0}^{N} b_i \cdot energy^i}$, to use the same methodology as for the energy calibration, the values for the resolution are squared and therefore the least-square fit of pairs $\left(energy_i, resolution_i^{\,2}\right), i = 1..no\ of\ reference\ peaks$ by the polynomial function $rer(channel)^2 = \sum_{i=0}^{N} b_i \cdot energy^i$ is used. The error is defined as the vector $\Delta\left(resolution^2\right) = 2 * \Delta resolution * resolution = \begin{pmatrix} 2 * \Delta resolution_1 * resolution_1 \\ 2 * \Delta resolution_2 * resolution_2 \\ ... \\ 2 * \Delta resolution_P * resolution_P \end{pmatrix}$.

In the pairs, $energy_i$ is the reference line energy and $resolution_i$ is the resolution as calculated by the initial peak search. N is the degree of the fitted polynomial function and it is configurable in the software. The $\Delta resolution$ is a resolution error calculated by the initial peak search.

The outputs of the least-square fitting are the vector of coefficients of the fitted polynomial function $B = \begin{pmatrix} b_0 \\ b_1 \\ ... \\ b_N \end{pmatrix}$ and the covariance matrix $\mathrm{cov}(B)$.

3.3.4.6. Competition

The purpose of the competition is to select the best one of the available sets of calibration coefficients. The competition among the sets of calibration coefficients is performed at the end of the calibration sequence. The "winning" set of coefficients is then selected for use in processing.

The competition is performed in two stages:

- Energy coefficients competition
- Resolution coefficients competition

There is a single selection algorithm that, for both ECR and RER, decides which calibration to select. This algorithm uses the same numerical procedures (least-square fit and statistical test) for both the ECR and the RER. The algorithm has three steps:

1. In the first step, the quality of the INITIAL coefficients is evaluated. The evaluation is based on the $F(\chi^2, n)$ distribution, with the χ^2 obtained in the polynomial fitting of the reference peaks from which the INITIAL coefficients has been determined. If the quality is below a defined threshold, the next two steps of the competition are not performed, i.e. no competition takes place. In this case the sample spectrum is considered unsuitable for calibration and calibration coefficients from the most recent prior (MRP) successful calibration of this detector are used, chosen from the prioritised list.
2. Only those ECR and RER candidates come into consideration, that do not show a significant shift relative to the ECR and RER relation from the current spectrum. The shift is evaluated using a shift test based on the $F(\chi^2, n)$ distribution, where χ^2 is calculated from the reference peaks energies, with energies calculated using candidate calibration coefficients and their uncertainties.
3. ECR and RER candidates are scored and the winners (one for ECR and one for RER) are selected for use in processing.

3.3.4.6.1. Energy competition

First, the command line parameters are evaluated:

- If the energy coefficients for the processing are specified on the command line, these are used and no other energy competition is performed.
- If the energy competition winner is specified on the command line, it is used and no other energy competition is performed.

If neither the coefficients nor the competition winner is specified as the command line parameter, the following algorithm applies:

INITIAL coefficients (calculated from the reference peaks found in the sample itself) are tested for quality sing the $F(\chi^2, n)$ distribution based on data from the polynomial fitting. The test is performed using the condition $F(\chi^2, n) < q$, where q is a configurable confidence level (default value 95%). If the condition is not met or the INITIAL coefficients are not available at all, the first available coefficients from a

prioritised list of (MRP_M, MRP_{QC}, MRP_A and INPUT in descending order of priority) are used and competition ends. Here

- MRP_M stands for the most recent sample spectrum from this detector that has undergone analyst review;
- MRP_{QC} stands for the most recent QC spectrum from this detector;
- MRP_A stands for the most recent sample spectrum from this detector that has undergone automated analysis; and
- INPUT stands for the coefficients in the message file of the current sample spectrum itself.

MRP coefficients are considered to be available if the corresponding MRP sample is found by a search query. INPUT coefficients are always available.

If the INITIAL coefficients pass the quality test, all candidates are tested for a possible shift, which, if present, would disqualify them. The shift is evaluated using the $F(\chi^2, n)$ distribution for each candidate coefficient set. First, the square of error is calculated for each peak:

$\Delta energy_i^2 = \sum_{k=1}^{M}\sum_{l=1}^{M} centroid\ channel_i^{(k-1)+(l-1)} \cdot \mathrm{cov}(k,l)$, where M is a degree of the candidate polynomial, $\mathrm{cov}(k,l)$ is an element from the covariance matrix and $channel_i$ is the centroid channel calculated by the reference peak search.

Then the least square is calculated:

$\chi^2 = \sum_{i=1}^{N} \frac{(energy(channel_i) - centroid\ energy_i)^2}{\Delta energy_i^2 + \Delta centroid\ energy^2}$, where N is the number of reference peaks, $energy()$ is the energy calculated using the polynomial defined by the candidate, $centroid\ channel_i$, $centroid\ energy_i$ and $\Delta centroid\ energy_i$ are the centroid channel, energy and energy error calculated by the reference peak search.

As the next step, the chi-square distribution $F(\chi^2, n)$ is calculated, where n is the degree of freedom and it equals to number of reference peaks.

Finally, the shift is tested using the condition $F(\chi^2, n) < q$, where q is a configurable confidence level (default value 95%). If the condition is satisfied, the candidate is qualified to enter the competition.

For the candidates that passed the shift test a score is calculated as $score = \max_{a \le channel \le b} \Delta energy(channel)$, where a and b are the channels at the energies defining the scoring energy range. These energies are configurable separately for particulate and Xenon samples.

The candidate with minimal score wins.

3.3.4.6.2. Resolution competition

The resolution competition is based on the winning energy coefficients from the energy competition and on reference peaks found in the peak search for resolution competition.

First, the command line parameters are evaluated:

- If the resolution coefficients for the processing are specified on the command line, these are used and no other resolution competition is performed.
- If the resolution competition winner is specified on the command line, it is used and no other resolution competition is performed.

If neither the coefficients nor the competition winner is specified as the command line parameter, the following algorithm applies. The algorithm is similar to the one used for energy competition:

INITIAL coefficients are tested for a shift using the $F(\chi^2, n)$ based on data from the polynomial fitting. The test is performed using the condition $F(\chi^2, n) < q$, where q is a configurable confidence level (default value 95%). If the condition fails or the INITIAL coefficients are not available at all, first available coefficients from MRP_M, MRP_{QC}, MRP_A and INPUT are used and competition ends.

If the test of INITIAL coefficients succeeds, all candidates are tested for a shift. Only the candidates passing the test will enter the competition. For each candidate, the following test applies:

The square of error is calculated for each peak:

$\Delta resolution_i^2 = \sqrt{\sum_{k=1}^{M}\sum_{l=1}^{M} centroid\ energy_i^{(k-1)+(l-1)} \cdot \mathrm{cov}(k,l)}$, where M is a degree of the candidate polynomial, $\mathrm{cov}(k,l)$ is an element from the covariance matrix and $centroid\ energy_i$ is the centroid energy calculated by the reference peak search.

Then the least square is calculated:

$\chi^2 = \sum_{i=1}^{N} \frac{(resolution(energy_i) - centroid\ resolution_i)^2}{\Delta resolution_i^2 + \Delta centroid\ resolution^2}$, where N is the number of reference peaks, $resolution()$ is the resolution calculated using the polynomial defined by the candidate, $centroid\ energy_i$, $centroid\ resolution_i$ and $\Delta centroid\ resolution_i$ are the centroid energy, resolution and resolution error calculated by the reference peak search.

As the next step, the chi-square distribution $F(\chi^2, n)$ is calculated, where n is the degree of freedom and it equals to number of reference peaks.

Finally, the shift is tested using the condition $F(\chi^2, n) < q$, where q is a configurable confidence level (default value 95%). If the condition is satisfied, the candidate is qualified to enter the competition.

For the candidates that passed the shift test the score is calculated as $score = \max_{a \le energy \le b} \Delta resolution(energy)$, where a and b are the channels at the energies defining the scoring energy range. These energies are configurable separately for particulate and Xenon samples.

The candidate with minimal score wins.

3.3.4.6.3. Run Efficiency Competition

First, the command line parameters are evaluated:

- If the efficiency coefficients for the processing are specified on the command line, they overwrite the calculated efficiency coefficients.
- If the VGSL pairs are available, they are used.
- If the VGSL pairs are not available, the coefficients are used.

3.3.4.7.Calculate Activities and Concentrations

For each peak and nuclide line found in the nuclides identification for particulate samples or for each relevant Xenon isotope line for Xenon samples, the following line characteristics are calculated.

$$Act_i = \frac{A_{ki} \cdot CCF \cdot DCF}{Y_i \cdot eff_{ki} \cdot T_L},$$

where

Act_i is an activity of the nuclide i

A_{ki} is a net key peak area

CCF is a coincidence correction factor

Y_i is a key line yield in the decay of nuclide i

eff_{ki} is detector efficiency at key line

T_L is an acquisition life time

and DCF is a decay correction factor.

$$DCF = \frac{\lambda \cdot T_S}{1 - e^{-\lambda \cdot T_S}} \cdot e^{\lambda \cdot T_D} \cdot \frac{\lambda \cdot T_A}{1 - e^{-\lambda \cdot T_A}},$$

where

T_S is a sampling time

T_A is an acquisition real time

T_D is a decay time

λ is a nuclide decay constant.

This formula assumes that the activity concentration in air stayed constant during sampling.

The relative uncertainty in the activity is calculated as the square root of the sum of the squares of the relative uncertainties in the area and the efficiency.

$$Act_{ierr} = \sqrt{A_{kierr}^2 + eff_{kierr}^2}$$

The nuclide concentration is derived from activity by dividing it with sampling volume:

$$Con_i = \frac{Act_i}{S_{vol}}$$

The data sources for the individual elements of the formula are described in the Table 4 (for particulate samples) and in the Table 5 (for Xenon samples)

Table 4 - Data sources for particulate activity calculation

Symbol	Data source
A_{ki}	Area of the key line peak from the peak search results. TPeak.area*TPeak.comment.->value, where TPeak.comment->keyFlag is set to 1 and TPeak.name == nuclide$_i$ name
A_{kierr}	Key line peak area uncertainty from the peak search results TPeak.areaUncertainty*TPeak.comment.->value where TPeak.comment->keyFlag is set to 1 and TPeak.name == nuclide$_i$ name.
CCF	**GARDS_IRF**.SUM_CORR
Y_i	**GARDS_NUCL_LINES_LIB**.ABUNDANCE
eff_{ki}	Efficiency of the key line peak from the peak search results. TPeak.efficiency.
eff_{kierr}	Efficiency uncertainty of the key line peak from the peak search results. TPeak.efficiencyUncertainty.
T_L	**GARDS_SAMPLE_DATA**.ACQUISITION_LIVE_SEC
T_S	(**GARDS_SAMPLE_DATA**.COLLECT_STOP-**GARDS_SAMPLE_DATA**.COLLECT_START) * 24 * 60 * 60
T_A	**GARDS_SAMPLE_DATA**.ACQUISITION_REAL_SEC

Symbol	Data source
T_D	(**GARDS_SAMPLE_DATA**.ACQUISITION_START-**GARDS_SAMPLE_DATA**.COLLECT_STOP) * 24 * 60 * 60
λ	$\lambda = \frac{\log(2)}{t}$, where t is half-life in seconds (**GARDS_NUCL_LIB**.HALFLIFE_SEC)
S_{vol}	**GARDS_SAMPLE_DATA**.QUANTITY

Table 5 - Data sources for Xenon activity calculation

Symbol	Data Source
A_{ki}	Area of the peak from the Xenon analysis results.
A_{kierr}	Area uncertainty from the Xenon analysis results
CCF	Constant "1".
Y_i	**GARDS_XE_NUCL_LINES_LIB**.ABUNDANCE
eff_{ki}	Detector efficiency at peak energy.
eff_{kierr}	1% of the detector efficiency.
T_L	**GARDS_SAMPLE_DATA**.ACQUISITION_LIVE_SEC
T_S	(**GARDS_SAMPLE_DATA**.COLLECT_STOP-**GARDS_SAMPLE_DATA**.COLLECT_START) * 24 * 60 * 60
T_A	**GARDS_SAMPLE_DATA**.ACQUISITION_REAL_SEC
T_D	(**GARDS_SAMPLE_DATA**.ACQUISITION_START-**GARDS_SAMPLE_DATA**.COLLECT_STOP) * 24 * 60 * 60
λ	$\lambda = \frac{\log(2)}{t}$, where t is half-life in seconds (**GARDS_XE:NUCL_LIB**.HALFLIFE_SEC)
S_{vol}	**GARDS_SAMPLE_AUX**.XE_VOLUME / 0.087

Minimum Detectable Activities (MDA) and Minimum Detectable Concentrations (MDC) of all nuclides of interest are calculated using the formulas below.

$$MDA_i = \frac{L_{DI} \cdot CCF \cdot DCF}{Y_i \cdot eff_{ki} \cdot T_L}$$

$$MDC_i = \frac{MDA_i}{S_{vol}}$$

$$L_{DI} = \frac{2 \cdot FWHMC_i \cdot (LCC_i - baseline_i)}{0.8591}$$

The data sources are described in the Table 6.

Table 6 - Data sources for minimum activity and concentration calculations

Symbol	Data Source
LCC_i	LCC at line channel
$baseline_i$	Baseline at line channel
$FWHMC_i$	FWHMC (channel resolution) at line channel

For particulate samples, the results are stored in **GARDS_NUCL_LINES_IDED** database table.

For Xenon samples, the results are stored in **GARDS_XE_RESULTS** database table.

In addition, decay uncorrected concentration is calculated for Xenon samples (with no DCF factor applied).

For particulate samples, nuclide characteristics are calculated out of line characteristics. Calculated characteristics include:

- Average activity
- Average activity uncertainty
- Activity at key line
- Activity uncertainty at key line
- Minimal MDA
- CSC ratio at key line
- CSC ratio uncertainty at key line
- CSC ratio flag at key line
- Nuclide found flag
- Decay correction factor

These results are stored in **GARDS_NUCL_IDED** database table.

3.4. Supporting Functions Library

3.4.1. Overview

The Supporting Functions Library contains functions used to prepare the data for the calculations and to parse the results of these calculations. This library is used by the Pipeline Wrapper.

The purpose of the library is to separate the database and file access from the calculations to keep the software modular and to allow for an easy introduction of additional calculations.

3.4.2. Dependencies

The Supporting Functions Library depends on the Infrastructure Library to perform logging and to access the configuration and on the Data Access Library to access the data. The interfaces to the library are defined by their respective header files. There is one interface function for each prepare-data and one for each parse-results high-level function defined in the section 3.4.4.

3.4.3. Requirements

There are no explicit requirements listed in [AUTO_SAINT_SRS] and [AUTO_XE_SAINT_SRS]. The design of the supporting functions is based on the existing source code and prototypes and on the results of discussions with the IDC.

Note: Additional requirements were identified and corresponding software changes designed and implemented in the test use of autoSaint.

3.4.4. Design decisions

The following high-level functions are defined in the Supporting Functions Library:

- Get calibration arrays
- Prepare data for and parse results of baseline calculation
- Prepare data for and parse results of SCAC calculation
- Prepare data for and parse results of LC calculation
- Prepare data for and parse results of the calibration and competition
- Prepare data for and parse results of the peak search
- Prepare data for and parse results of nuclide identification
- Prepare data for and parse results of Xenon analysis

3.4.4.1. Get Calibration Arrays

The calibration is based on Most Recent Prior (MRP) values. It will use the MRP values during the calibration and it will update them based on the calibration results. The following MRP values are used in *autoSaint*:

- MRP_A: processing parameters from the previous automatic processing for that particular station, detector and sample type.
- MRP_M: processing parameters from the previous manual processing. The processing parameters are reading from Manual database source. The manual data source name, user name and password are read from the command line or the database.
- MRP_{QC}: processing parameters from the previous automatic processing for that particular station, detector and QC sample.
- INPUT: processing parameters calculated from the pairs defined in the sample itself.
- INITIAL: processing parameters calculated from the reference peaks identified in the sample itself.

It is possible to override the processing parameters by specifying them in the software configuration.

The energy and calibration arrays are being prepared using the MRP_A regression coefficients, if they are available. If they are not available, INPUT regression coefficients calculated from the input pairs are used.

Details of the calculations are defined in section 3.3.4.4.

SQL queries used to read MRPs:

```
SELECT GSD.SAMPLE_ID FROM GARDS_SAMPLE_DATA GSD, GARDS_SAMPLE_STATUS GSS WHERE
GSD.ACQUISITION_START < (SELECT ACQUISITION_START FROM GARDS_SAMPLE_DATA WHERE SAMPLE_ID =
%sampleId) AND GSD.DATA_TYPE = '%dataType' AND GSD.DETECTOR_ID = %detectorId AND
GSD.SAMPLE_TYPE = '%sampleType' AND GSD.SPECTRAL_QUALIFIER = '%spectralQualifier' AND
GSD.SAMPLE_ID = GSS.SAMPLE_ID AND GSS.STATUS IN ('P', 'R', 'Q', 'V') AND NOT (GSS.STATUS =
'Q' AND CATEGORY IS NULL) ORDER BY ACQUISITION_START DESC

SELECT COEFF1, COEFF2, COEFF3, COEFF4, COEFF5, COEFF6, COEFF7, COEFF8 FROM GARDS_ENERGY_CAL
WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT COEFF1, COEFF2, COEFF3, COEFF4, COEFF5, COEFF6, COEFF7, COEFF8 FROM
GARDS_RESOLUTION_CAL WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT COEFF1, COEFF2, COEFF3, COEFF4, COEFF5, COEFF6, COEFF7, COEFF8 FROM
GARDS_EFFICIENCY_CAL WHERE SAMPLE_ID = %sampleId
```

3.4.4.2. Prepare Data For and Parse Results of Baseline Calculation

Sample spectrum, energy array and the resolution array are prepared for the baseline calculation. The sample spectrum is read from the sample file and energy and resolution arrays are calculated as defined in section 3.4.4.1.

If configured, the inputs are stored as intermediate data files.

After the baseline calculation, the baseline is stored in the file based store and entries are created in the SQL database which point to the baseline file. The location of the file store is configurable.

SQL queries used to read baseline configuration:

```
SELECT INDEX_NO, CAST(NVL(ENERGY_LOW,-1.0) AS NUMBER), CAST(NVL(ENERGY_HIGH,-1.0) AS
NUMBER), MULT, NO_OF_LOOPS FROM GARDS_BASELINE WHERE DETECTOR_ID = %detectorId AND
DATA_TYPE = '%dataType'  ORDER BY INDEX_NO

SELECT CAST(INDEX_NO AS NUMBER(11,1)), CAST(NVL(ENERGY_LOW,-1.0) AS NUMBER),
CAST(NVL(ENERGY_HIGH,-1.0) AS NUMBER), CAST(MULT AS NUMBER), NO_OF_LOOPS FROM
GARDS_BASELINE WHERE DETECTOR_ID IS NULL AND DATA_TYPE = '%dataType' AND SAMPLE_TYPE =
'%sampleType' ORDER BY INDEX_NO
```

SQL queries used when storing results:

```
SELECT CAST(TYPEID AS NUMBER(11,1)) FROM GARDS_PRODUCT_TYPE WHERE PRODTYPE = 'BASELINE'

SELECT CAST(NVL(MAX(REVISION), 0) AS NUMBER(11,1))  FROM GARDS_PRODUCT WHERE SAMPLE_ID =
%sampleId AND TYPEID = %productTypeId

DELETE GARDS_PRODUCT WHERE SAMPLE_ID = %sampleId AND TYPEID = %productTypeId

INSERT INTO GARDS_PRODUCT (SAMPLE_ID, FOFF, DSIZE, DIR, DFILE, REVISION, TYPEID, AUTHOR,
MODDATE) VALUES (%sampleId, %offset, %size, '%path', '%filename', %revisionNumber, %typeId,
'auto_SAINT', to_date('%currentDateTime', 'YYYY-MM-DD HH24:MI:SS'))
```

3.4.4.3. Prepare Data for and Parse Results of SCAC Calculation

Sample spectrum and the resolution array are prepared for the SCAC calculation in the same way as for the baseline calculation described in section 3.4.4.2.

If configured, the inputs are stored as intermediate data files.

After the SCAC calculation, the SCAC is stored in the file based store and entries will be created in the SQL database which point to the SCAC file.

SQL queries used when storing results:

```
SELECT CAST(TYPEID AS NUMBER(11,1)) FROM GARDS_PRODUCT_TYPE WHERE PRODTYPE = 'SCAC'

SELECT CAST(NVL(MAX(REVISION), 0) AS NUMBER(11,1))  FROM GARDS_PRODUCT WHERE SAMPLE_ID =
%sampleId AND TYPEID = %productTypeId

DELETE GARDS_PRODUCT WHERE SAMPLE_ID = %sampleId AND TYPEID = %productTypeId

INSERT INTO GARDS_PRODUCT (SAMPLE_ID, FOFF, DSIZE, DIR, DFILE, REVISION, TYPEID, AUTHOR,
MODDATE) VALUES (%sampleId, %offset, %size, '%path', '%filename', %revisionNumber,
%productTypeId, 'auto_SAINT', to_date('%currentDateTime', 'YYYY-MM-DD HH24:MI:SS'))
```

3.4.4.4. Prepare Data For and Parse Results of LC Calculation

Sample spectrum, baseline and the resolution array are prepared for the LC calculation the same way as for the baseline calculation described in section 3.4.4.2.

3.4.4.5. Prepare Data For and Parse Results of Calibration and Competition

Inputs required by calibration and competition algorithms are prepared.

SQL queries used in calibration and competition:

```
SELECT GARDS_SAMPLE_DATA.SAMPLE_ID FROM GARDS_SAMPLE_AUX, GARDS_SAMPLE_DATA WHERE
(GARDS_SAMPLE_DATA.SAMPLE_ID=GARDS_SAMPLE_AUX.SAMPLE_ID) AND (SAMPLE_REF_ID = (SELECT
SAMPLE_REF_ID FROM GARDS_SAMPLE_AUX WHERE (SAMPLE_ID = %sampleId))) AND
(GARDS_SAMPLE_DATA.SAMPLE_ID <> %sampleId) AND (DETECTOR_ID = %detectorId) ORDER BY
GARDS_SAMPLE_DATA.ACQUISITION_STOP

SELECT REFPEAK_ENERGY FROM GARDS_REFLINE_MASTER WHERE DATA_TYPE = '%dataType' AND
SPECTRAL_QUALIFIER = '%spectralQualifier' AND (CALIBRATION_TYPE='%calibrationType' OR
CALIBRATION_TYPE IS NULL) ORDER BY REFPEAK_ENERGY

SELECT REFPEAK_ENERGY FROM GARDS_XE_REFLINE_MASTER WHERE DATA_TYPE = '%dataType' AND
SPECTRAL_QUALIFIER = '%spectralQualifier' AND (CALIBRATION_TYPE='%calibrationType' OR
CALIBRATION_TYPE IS NULL) ORDER BY REFPEAK_ENERGY

SELECT EFFIC_ENERGY, EFFICIENCY, EFFIC_ERROR FROM GARDS_EFFICIENCY_VGSL_PAIRS WHERE
DETECTOR_ID = %detectorId AND BEGIN_DATE<=to_date('%acquisitionStart','YYYY/MM/DD
HH24:MI:SS') AND END_DATE>to_date('%acquisitionStop','YYYY/MM/DD HH24:MI:SS') ORDER BY
EFFIC_ENERGY

SELECT CAST(ROW_INDEX AS NUMBER(11,1)), CAST(COL_INDEX AS NUMBER(11,1)), COEFF FROM
GARDS_ENERGY_CAL_COV WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT CAST(ROW_INDEX AS NUMBER(11,1)), CAST(COL_INDEX AS NUMBER(11,1)), COEFF FROM
GARDS_RESOLUTION_CAL_COV WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT CHANNEL, CAL_ENERGY, CAL_ERROR FROM GARDS_ENERGY_PAIRS WHERE SAMPLE_ID = %sampleId
AND (WINNER='Y' OR WINNER IS NULL)

SELECT CHANNEL, CAL_ENERGY, CAL_ERROR FROM GARDS_ENERGY_PAIRS_ORIG WHERE SAMPLE_ID =
%sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT RES_ENERGY, RESOLUTION, RES_ERROR FROM GARDS_RESOLUTION_PAIRS WHERE SAMPLE_ID =
%sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT RES_ENERGY, RESOLUTION, RES_ERROR FROM GARDS_RESOLUTION_PAIRS_ORIG WHERE SAMPLE_ID =
%sampleId AND (WINNER='Y' OR WINNER IS NULL)

SELECT EFFIC_ENERGY, EFFICIENCY, EFFIC_ERROR FROM GARDS_EFFICIENCY_PAIRS WHERE SAMPLE_ID =
%sampleId

DELETE GARDS_ENERGY_CAL WHERE SAMPLE_ID=%sampleId

DELETE GARDS_ENERGY_CAL_COV WHERE SAMPLE_ID=%sampleId

DELETE GARDS_RESOLUTION_CAL WHERE SAMPLE_ID=%sampleId

DELETE GARDS_RESOLUTION_CAL_COV WHERE SAMPLE_ID=%sampleId

DELETE GARDS_EFFICIENCY_CAL WHERE SAMPLE_ID=%sampleId

INSERT INTO GARDS_ENERGY_CAL (SAMPLE_ID, COEFF1, COEFF2, COEFF3, COEFF4, COEFF5, COEFF6,
COEFF7, COEFF8, ENERGY_UNITS, CNV_FACTOR, APE, DET, MSE, TSTAT, SCORE, TYPE, WINNER) VALUES
( %sampleId, %c1, %c2, %c3, %c4, %c5, %c6, %c7, %c8, ‘’, -1, -1, -1, -1, -1, %score,
'%type', '%winnerFlag')

INSERT INTO GARDS_RESOLUTION_CAL (SAMPLE_ID, COEFF1, COEFF2, COEFF3, COEFF4, COEFF5,
COEFF6, COEFF7, COEFF8, TYPE, WINNER) VALUES ( %sampleId, %c1, %c2, %c3, %c4, %c5, %c6,
%c7, %c8, '%type', '%winnerFlag')

INSERT INTO GARDS_EFFICIENCY_CAL (SAMPLE_ID, DEGREE, EFFTYPE, COEFF1, COEFF2, COEFF3,
COEFF4, COEFF5, COEFF6, COEFF7, COEFF8) VALUES ( %sampleId, %polyDegree, '%vgslOrEmp', %c1,
%c2, %c3, %c4, %c5, %c6, %c7, %c8)

INSERT INTO GARDS_ENERGY_CAL_COV (SAMPLE_ID, ROW_INDEX, COL_INDEX, COEFF, TYPE, WINNER)
VALUES (%sampleId, %row, %col, %coeff, '%type', '%winnerFlag')

INSERT INTO GARDS_RESOLUTION_CAL_COV (SAMPLE_ID, ROW_INDEX, COL_INDEX, COEFF, TYPE, WINNER)
VALUES (%sampleId, %row, %col, %coeff, '%type', '%winnerFlag')

DELETE FROM GARDS_ENERGY_PAIRS WHERE SAMPLE_ID = %sampleId AND TYPE='%type'

DELETE FROM GARDS_ENERGY_PAIRS WHERE SAMPLE_ID = %sampleId AND (TYPE='%type' OR TYPE IS
NULL)

DELETE FROM GARDS_RESOLUTION_PAIRS WHERE SAMPLE_ID = %sampleId AND TYPE='%type'

DELETE FROM GARDS_RESOLUTION_PAIRS WHERE SAMPLE_ID = %sampleId AND (TYPE='%type' OR TYPE IS
NULL)
```

```
INSERT INTO GARDS_ENERGY_PAIRS (SAMPLE_ID, CAL_ENERGY, CHANNEL, CAL_ERROR, TYPE, WINNER)
VALUES (%sampleId, %energy, %channel, %energyUncertainty, 'INITIAL', 'N')

INSERT INTO GARDS_RESOLUTION_PAIRS (SAMPLE_ID, RES_ENERGY, RESOLUTION, RES_ERROR, TYPE,
WINNER) VALUES (%sampleId, %energy, %resolution, %resolutionUncertainty, 'INITIAL', 'N')

INSERT INTO GARDS_ENERGY_PAIRS (SAMPLE_ID, CAL_ENERGY, CHANNEL, CAL_ERROR, TYPE, WINNER)
(SELECT SAMPLE_ID, CAL_ENERGY, CHANNEL, CAL_ERROR, 'INPUT', '%winnerFlag' FROM
GARDS_ENERGY_PAIRS_ORIG WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL))

INSERT INTO GARDS_RESOLUTION_PAIRS (SAMPLE_ID, RES_ENERGY, RESOLUTION, RES_ERROR, TYPE,
WINNER) (SELECT SAMPLE_ID, RES_ENERGY, RESOLUTION, RES_ERROR, 'INPUT', '%winnerFlag' FROM
GARDS_RESOLUTION_PAIRS_ORIG WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL))

INSERT INTO GARDS_ENERGY_PAIRS (SAMPLE_ID, CAL_ENERGY, CHANNEL, CAL_ERROR, TYPE, WINNER)
(SELECT %d, CAL_ENERGY, CHANNEL, CAL_ERROR, '%type', '%winnerFlag' FROM GARDS_ENERGY_PAIRS
WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL))

INSERT INTO GARDS_ENERGY_PAIRS (SAMPLE_ID, CAL_ENERGY, CHANNEL, CAL_ERROR, TYPE, WINNER)
(SELECT %d, CAL_ENERGY, CHANNEL, CAL_ERROR, '%type', '%winnerFlag' FROM
%manAccount.GARDS_ENERGY_PAIRS WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS
NULL))

INSERT INTO GARDS_RESOLUTION_PAIRS (SAMPLE_ID, RES_ENERGY, RESOLUTION, RES_ERROR, TYPE,
WINNER) (SELECT %sampleId, RES_ENERGY, RESOLUTION, RES_ERROR, '%type', '%winnerFlag' FROM
GARDS_RESOLUTION_PAIRS WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS NULL))

INSERT INTO GARDS_RESOLUTION_PAIRS (SAMPLE_ID, RES_ENERGY, RESOLUTION, RES_ERROR, TYPE,
WINNER) (SELECT %sampleId, RES_ENERGY, RESOLUTION, RES_ERROR, '%type', '%winnerFlag' FROM
%manAccount.GARDS_RESOLUTION_PAIRS WHERE SAMPLE_ID = %sampleId AND (WINNER='Y' OR WINNER IS
NULL))
```

3.4.4.6. Prepare Data for and Parse Results of Peak Search

Sample spectrum, energy array, resolution (in channels and energy) arrays, baseline, SCAC and LC arrays and efficiency coefficients are prepared for peak search.

SQL queries used when storing results:

```
DELETE FROM GARDS_PEAKS WHERE SAMPLE_ID = %sampleId

INSERT INTO GARDS_PEAKS (SAMPLE_ID, PEAK_ID, CENTROID, CENTROID_ERR, ENERGY, ENERGY_ERR,
LEFT_CHAN, WIDTH, BACK_COUNT, BACK_UNCER, FWHM, FWHM_ERR, AREA, AREA_ERR, ORIGINAL_AREA,
ORIGINAL_UNCER, COUNTS_SEC, COUNTS_SEC_ERR, EFFICIENCY, EFF_ERROR, BACK_CHANNEL, IDED,
FITTED, MULTIPLET, PEAK_SIG, LC, PSS, DETECTABILITY) VALUES (%sampleId, %peakId, %centroid,
%centroidUncertainty, %energy, %energyUncertainty, %leftChannel, %width, %bkgndCounts,
%bkgndCountsUncertainty, %fwhm, %fwhmUncertainty, %area, %areaUncertainty, NULL, NULL,
%counts, %countsUncertainty, %efficiency, %efficiencyUncertainty, %backChannel, %idedFlag,
NULL, NULL, NULL, %lc, NULL, %detectability)
```

3.4.4.7. Prepare Data for and Parse Results of Nuclides Identification and Particulate Activities Calculations

In addition to the inputs and the results of the peak search, nuclide characteristics are loaded from the database before the nuclide identification.

The results are stored in the database tables **GARDS_NUCL_LINES_IDED** and **GARDS_NUCL_IDED**. The entries generated during nuclide identification are later updated in the activities calculation step.

SQL queries used to prepare for calculations:

```
SELECT ENERGY, IRF, IRF_ERROR, SUM_CORR FROM GARDS_IRF WHERE DETECTOR_ID = %detectorId AND
NUCLIDE_NAME = '%nuclideName' AND BEGIN_DATE<=to_date('%acquisitionStart', 'YYYY/MM/DD
HH24:MI:SS') AND END_DATE>to_date('%acquisitionStop','YYYY/MM/DD HH24:MI:SS')

SELECT NAME, NID_FLAG FROM GARDS_NUCL_IDED WHERE SAMPLE_ID = %sampleId

SELECT ACTIVITY, ACTIV_ERR, MDA, KEY_FLAG, CSC_RATIO, CSC_RATIO_ERR, CSC_MOD_FLAG,
NUCLIDE_ID FROM GARDS_NUCL_LINES_IDED WHERE SAMPLE_ID = %sampleId AND NAME = '%nuclideName'

SELECT NAME, ENERGY, ABUNDANCE, KEY_FLAG FROM GARDS_NUCL_LINES_LIB WHERE ABUNDANCE!=0

SELECT ENERGY_ERR, ABUNDANCE, ABUNDANCE_ERR, ENERGY FROM GARDS_NUCL_LINES_LIB WHERE NAME =
'%nuclideName' AND ENERGY BETWEEN %energyLow AND %energyHigh

SELECT ENERGY_ERR, ABUNDANCE, ABUNDANCE_ERR, ENERGY FROM GARDS_NUCL_LINES_LIB WHERE NAME =
'%nuclideName' AND KEY_FLAG = 1

SELECT NUCLIDE_ID, TYPE, HALFLIFE, HALFLIFE_SEC FROM GARDS_NUCL_LIB WHERE NAME =
'%nuclideName'
```

SQL queries used when storing results:

```
UPDATE GARDS_PEAKS SET IDED = 1 WHERE (SAMPLE_ID = %sampleId) AND (PEAK_ID IN (SELECT
DISTINCT PEAK FROM GARDS_NUCL_LINES_IDED WHERE (SAMPLE_ID = %sampleId)))

DELETE FROM GARDS_NUCL_LINES_IDED WHERE SAMPLE_ID = %sampleId

INSERT INTO GARDS_NUCL_LINES_IDED (SAMPLE_ID, STATION_ID, DETECTOR_ID, NAME, ENERGY,
ENERGY_ERR, ABUNDANCE, ABUNDANCE_ERR, PEAK, ACTIVITY, ACTIV_ERR, EFFIC, EFFIC_ERR, MDA,
KEY_FLAG, NUCLIDE_ID, CSC_RATIO, CSC_RATIO_ERR, CSC_MOD_FLAG, ID_PERCENT ) VALUES (
%sampleId, %stationId, %detectorId, '%nuclideName', %energy, %energyUncertainty,
%abundance, %abundanceUncertainty, %peak, %activity, %activityUncertainty, %efficiency,
%efficiencyUncertainty, %mda, %keyFlag, %nuclideId, %cscRatio, %cscRatioUncertainty,
%cscModFlag, %idPercent )

DELETE FROM GARDS_NUCL_IDED WHERE SAMPLE_ID = %sampleId

INSERT INTO GARDS_NUCL_IDED (SAMPLE_ID, STATION_ID, DETECTOR_ID, NAME, NID_FLAG) ( SELECT
DISTINCT SAMPLE_ID, STATION_ID, DETECTOR_ID, NAME, 1 FROM GARDS_NUCL_LINES_IDED WHERE
SAMPLE_ID = %sampleId )

INSERT INTO GARDS_NUCL_IDED (SAMPLE_ID, STATION_ID, DETECTOR_ID, NAME, NID_FLAG) (SELECT
%sampleId, %stationId, %detectorId, NAME, 0 FROM GARDS_NUCL_LIB WHERE GARDS_NUCL_LIB.TYPE
NOT LIKE 'FISSION (G)' AND GARDS_NUCL_LIB.TYPE NOT LIKE 'FISSION(G)' AND NAME NOT IN
(SELECT DISTINCT NAME FROM GARDS_NUCL_IDED WHERE SAMPLE_ID = %sampleId ))

UPDATE GARDS_NUCL_IDED SET NUCLIDE_ID = %nuclideId, TYPE = '%nuclideType', HALFLIFE =
'%halflife', AVE_ACTIV = %averageActivity, AVE_ACTIV_ERR = %averageActivityUncertainty,
ACTIV_KEY = %keyLineActivity, ACTIV_KEY_ERR= %keyLineActivityUncertainty, MDA = %mda,
CSC_RATIO = %cscRatio, CSC_RATIO_ERR = %cscRatioUncertainty, CSC_MOD_FLAG = %cscModFlag,
PD_MOD_FLAG = %pdModFlag, ACTIV_DECAY_ERR = 0, NID_FLAG = %nidFlag, ACTIV_DECAY =
%activityDecay, REPORT_MDA = ( SELECT COUNT(*) FROM GARDS_MDAS2REPORT WHERE NAME =
GARDS_NUCL_IDED.NAME AND SAMPLE_TYPE = '%sampleType' AND DTG_BEGIN <
to_date('%acquisitionStart', 'YYYY-MM-DD HH24:MI:SS') AND ( DTG_END >
to_date('%acquisitionStop', 'YYYY-MM-DD HH24:MI:SS') OD DTG_END IS NULL) ) WHERE SAMPLE_ID
= %sampleId AND NAME = '%nuclideName'
```

3.4.4.8. Prepare Data for and Parse Results of Xenon Calculations and Xenon Activities Calculations

Energy array, resolution array, efficiency coefficients and/or pairs, the description of Xenon isotopes and full and preliminary samples data are prepared for Xenon analysis.

The results of the analysis are stored in the database table **GARDS_XE_RESULTS**.

SQL queries used to prepare for calculations:

```
SELECT NAME, ENERGY, ABUNDANCE, KEY_FLAG FROM GARDS_XE_NUCL_LINES_LIB WHERE ABUNDANCE!=0

SELECT ENERGY_ERR, ABUNDANCE, ABUNDANCE_ERR, ENERGY FROM GARDS_XE_NUCL_LINES_LIB WHERE NAME
= '%nuclideName' AND ENERGY BETWEEN %energyLow AND %energyHigh

SELECT ENERGY_ERR, ABUNDANCE, ABUNDANCE_ERR, ENERGY FROM GARDS_XE_NUCL_LINES_LIB WHERE NAME
= '%nuclideName' AND KEY_FLAG = 1

SELECT NUCLIDE_ID, TYPE, HALFLIFE, HALFLIFE_SEC FROM GARDS_XE_NUCL_LIB WHERE NAME =
'%nuclideName'

SELECT ABUNDANCE FROM GARDS_XE_NUCL_LINES_LIB WHERE NAME='%nuclideName' AND ENERGY BETWEEN
%energyLow AND %energyHigh

SELECT ACTIV_DECAY FROM GARDS_NUCL_IDED WHERE SAMPLE_ID=%sampleId AND NAME='%nuclideName'

SELECT ENERGY, ABUNDANCE FROM GARDS_XE_NUCL_LINES_LIB WHERE NAME='%nuclideName' ORDER BY
ENERGY
```

SQL queries used when storing results:

```
DELETE GARDS_XE_RESULTS WHERE SAMPLE_ID=%sampleId

DELETE GARDS_XE_UNCORRECTED_RESULTS WHERE SAMPLE_ID=%sampleId

DELETE GARDS_XE_RESULTS WHERE SAMPLE_ID=%sampleId AND METHOD_ID=%methodId

DELETE GARDS_XE_UNCORRECTED_RESULTS WHERE SAMPLE_ID=%sampleId AND METHOD_ID=%methodId

INSERT INTO GARDS_XE_RESULTS (SAMPLE_ID,METHOD_ID,NUCLIDE_ID,CONC,CONC_ERR,MDC,MDI,
NID_FLAG,LC,LD,SAMPLE_ACT,COV_XE_131M,COV_XE_133M,COV_XE_133,COV_XE_135,COV_RADON) VALUES
(%sampleId,%methodId,%nuclideId,%concentration,%concentrationUncertainty,NULL,%mdi,%nidFlag
,%lc,%ld,%activity,%covXe131m,%cpvXe133m,%covXe133,%covXe135,%covRadon)

INSERT INTO GARDS_XE_UNCORRECTED_RESULTS (SAMPLE_ID, METHOD_ID, NUCLIDE_ID, CONC,
CONC_ERROR, MDC, MDI, LC) VALUES
(%sampleId,%methodId,%nuclideId,%concentration,%concentrationUncertainty,NULL,%mdi,%lc)

INSERT INTO GARDS_XE_RESULTS (SAMPLE_ID, METHOD_ID, NUCLIDE_ID, CONC, CONC_ERR, MDC, MDI,
NID_FLAG,LC,LD,SAMPLE_ACT,COV_XE_131M,COV_XE_133M,COV_XE_133,COV_XE_135,COV_RADON) VALUES
(%sampleId,%methodId,%nuclideId,%concentration,%concentrationUncertainty,%mdc,NULL,%nidFlag
,%lc,%ld,%activity,%covXe131M,%covXe133M,%covXe133,%covXe135,%covRadon)

INSERT INTO GARDS_XE_UNCORRECTED_RESULTS (SAMPLE_ID, METHOD_ID, NUCLIDE_ID, CONC,
CONC_ERROR, MDC, MDI, LC) VALUES
(%sampleId,%methodId,%nuclideId,%concentration,%concentrationUncertainty,%mdc,NULL,%lc)
```

3.5. Infrastructure Library

3.5.1. Overview

The Infrastructure Library contains functions used to access the software configuration, write log entries and handle errors. This library is used by all other components of the *autoSaint* software.

3.5.2. Dependencies

The Infrastructure Library depends on the Data Access Library to read the configuration entries defined in the SQL database. The interface to the library is defined by its respective header file.

3.5.3. Requirements

Table 7 – Requirements allocated to Infrastructure Library

Requirement	Addressed by
The software shall flush the write buffer every time a message is written to the log file.	3.5.4.2
The software shall have the capability to send all log messages to the standard UNIX facility Syslog.	3.5.4.2
The Syslog functionality shall meet the standards defined in [IDC_SYSLOG_2003].	3.5.4.2
The software shall have the capability to send the Syslog messages to standard output.	3.5.4.2
The software shall log the date and time, to the nearest second, the software was started.	3.5.4.2
The software shall display a descriptive list of all possible parameters if it is started with a – "h" parameter.	3.5.4.1
The number of hard-coded parameters should be reduced to a minimum.	Appendix II CONFIGURATION PARAMETERS
The user shall be able to operate and control the software via any combination of the following: (a) Command line parameters; (b) Parameters in the database.	3.5.4.1
Each configurable parameter shall have a default value.	Appendix II CONFIGURATION PARAMETERS
The software shall allow a set of default values to be defined for each detector.	Software uses the configuration defined in the database.

Requirement	Addressed by
The software shall log details of each error or warning raised by the application, including: (a) The Sample identifier, (b) The reason for the error or warning. The software shall attempt to log the date, the time to the nearest second, and the reason the software was terminated. It is expected that in some situations it will be impossible for the software to log this information, for example, if a Unix kill –9 is sent.	3.5.4.2
The software shall allow the user to specify a debug level between 0 and 9. Syslog messages shall be unaffected by the debug level.	3.5.4.2, Table 8
If the debug level is 0, then only start-up and close-down messages shall be sent to standard output.	3.5.4.2, Table 8
If the debug level is 1 then all Syslog messages shall also be sent to standard output.	3.5.4.2, Table 8
If the debug level is between 2 and 9 inclusive then additional debug messages shall be sent to standard output (where 9 provides the maximum volume of debug messages).	3.5.4.2, Table 8
The debug levels used should mirror the debug levels used in the *bg_analyze* software as closely as possible.	3.5.4.2, Table 8

Note: Additional requirements were identified and corresponding software changes designed and implemented in the test use of autoSaint.

3.5.4. Design decisions

3.5.4.1. Software Configuration

The Infrastructure Library implements the functions used to read the software configuration. There are three places where the configuration can be defined: command line parameters, database entries (**GARDS_SAINT_DEFAULT_PARAMS** table) and default values defined in the *autoSaint* software.

The search for the configuration items is performed in the following order (first occurrence wins):

- command line parameters
- configuration items that are stored in the database
- default values (defined in the source code)

If requested by the command line parameter, the list of all configuration attributes and their description is displayed and the software exits.

3.5.4.2. Logging

The *autoSaint* software writes two types of log entries.

The syslog entries uses the syslog libraries to write entries of the types ("`err`", "`warning`", "`notice`" and "`info`"). The Syslog functionality meets the standards defined in [IDC_SYSLOG_2003].

The syslog messages contain the sample ID to link the message to the processed sample.

The debug log entries write messages to the standard error console. The level of logging is configurable from 0 to 9. The software flushes the standard error output each time the message is written to make sure that all messages are stored. This mechanism safeguards against messages being lost due to software failure (i.e. crashing). The syslog messages are also written to the debug log entries if a high enough log level is used.

The log messages contain the timestamp, source of the message, message text and message details.

Table 8 contains descriptions of log event types and the minimum log levels required to write the log messages to that particular event type.

Table 8 – Logging verbosity

Event Type	Log Level
`START_STOP`: An application start or stopping message.	This message is always written to the system log and will be written to the debug log if the log level >= 0.
`ERROR`: An application error message.	This message is always written to the system log and will be written to the debug log if the log level >= 1.
`WARNING`: An application warning message.	This message is always written to the system log and will be written to the debug log if the log level >= 1.
`ANALYST_INFO:` Processing intermediate data and results.	This message is written to the debug log if the log level >= 2.
`PROCESSING_PARAMETERS`: A message regarding the value of a processing parameter.	This message is written to the debug log if the log level >= 3.
`CONTROL_FLOW:` A message indicating when a function is entered and left.	This message is written to the debug log if the log level >= 6.
`DB_ACCESS:` Executed SQL queries.	This message is written to the debug log if the log level >= 5.
`QC_ROUTINE_FAILURE:` Warnings on QC routines failures.	This message is always written to the system log and will be written to the debug log if the log level >= 1.
`CALIBRATION:` Sample calibration.	This message is written to the debug log if the log level >= 4.

4. INTERFACE ENTITIES

4.1. Data Access

4.1.1. Overview

The *autoSaint* software accesses the file based data store and the SQL database. The file based store is managed using the IDC **FPDESCRIPTION** and **FILEPRODUCT** database tables. The SQL database is accessed using the gODBC library provided by IDC.

4.1.2. Dependencies

The data access depends on the gODBC library to access the SQL database and on **FPDESCRIPTION** and **FILEPRODUCT** database tables to access the file based data store.

4.1.3. Requirements

Table 9 – Requirements allocated to Data Access Layer

Requirement	Addressed by
If a file operation fails (e.g. open, read, write, seek, close) then the software shall generate an error and terminate.	0
If the software is unable to write, to the database, any of the intermediate or final results then the software shall generate an error message and terminate.	0
The software shall only write results from sample processing to the database if the processing of the sample completes successfully.	0
The software shall only access the database through Open Database Connectivity (ODBC).	0
The software shall only access ODBC through the ‘gODBC’ library (provided free-of-charge as part of gbase-1.1.9 by IDC).	0
The user shall have read and write access to all files written by the software. By default, the software shall grant read privileges to all users.	0

Note: Additional requirements were identified and corresponding software changes designed and implemented in test use of autoSaint.

4.1.4. Design Decisions

There is no dedicated module for data access. The data access is a part of the Supporting Functions and Infrastructure libraries.

The following common attributes apply to all data access calls:

- The file based data store is accessed through the IDC **FPDESCRIPTION** and **FILEPRODUCT** tables. The access mask to the generated files will is configurable. By default, the software will grant read privileges to all users.
- The SQL database is accessed through the gODBC library provided by IDC.
- Functions that require data access take responsibility for error handing. A data access error will result in a managed termination of the application.
- There is only one connection to the SQL database per instance of the application. This connection will be opened during the initialization and closed at the end of the processing. In the case of successful processing, the SQL transaction will be committed. When an error occurs the SQL transaction will be rolled back.

APPENDIX I
ADDITIONAL REQUIREMENTS

Table 10 – Requirements not allocated to specific software components

Requirement	Addressed by
Security requirement: The software will contain only the functionality described in this document. The software will not contain any additional functionality.	The Contractor has implemented the software as defined in this document and as requested in customer defined change requests.
The software shall be able to complete the automatic processing for a single sample in 1 minute or less.	The software design described in this document and its implementation have attempted to implement a performance-effective solution to the processing of samples.
The software shall be able to automatically process 1000 sets of sample data per day.	The software design described in this document and its implementation have attempted to implement a performance-effective solution to the processing of samples.
The software shall be able to process 100 samples, under typical operational conditions, without any crashes or memory leaks (with the exception of memory leaks from third party libraries).	The Contractor has applied the quality practices in the software development as defined in the [AUTO_SAINT_QP]. While no software practice can completely eliminate the risk of crashes and memory leaks in C language programs, it reduces it significantly.
The software, with the exception of third-party libraries, shall have no memory leaks.	The Contractor has applied the quality practices in software development as defined in the [AUTO_SAINT_QP]. While no software practice can completely eliminate the risk of memory leaks in C language programs, it reduces it significantly.
The software shall meet IDC documentation standards.	This requirement does not affect the design or the implementation of the software.
All user documentation shall be written in English.	This requirement does not affect the design or the implementation of the software.
All user documentation shall follow the requirements specified in the IDC Corporate Identity Style Manual (2002).	This requirement does not affect the design or the implementation of the software.
A user manual shall be provided that follows the [IDC_SUT_2003]. The user manual format and structure should be as close as possible to the [BG_ANALYZE_SUT].	This requirement does not affect the design or the implementation of the software.
There shall be a full set of man pages describing how to use the system.	This requirement does not affect the design of the software.

Requirement	Addressed by
The user documentation shall stress that the process that performs the automatic processing leading to an Automatic Radionuclide Report (ARR) should only access the Auto database.	This requirement does not affect the design or the implementation of the software.
The user documentation shall stress that the process that performs the automatic processing leading to a Reviewed Radionuclide Report (RRR) should only access the Man database.	This requirement does not affect the design or the implementation of the software.
A design shall be provided that follows the IDC Software Design Description Template (2003).	This document conforms to the IDC Software Design Description Template (2003).
A software acceptance test plan shall be provided that follows the IDC Software Test Plan Template (2003).	This requirement does not affect the design or the implementation of the software.
A software acceptance test description shall be provided that follows the IDC Software Test Description Template (2003).	This requirement does not affect the design or the implementation of the software.
The software acceptance test plan format and structure should be as close as possible to the ‘Bg_analyze software acceptance test plan’ (IDC, 2005)	This requirement does not affect the design or the implementation of the software.
The software acceptance test description format and structure should be as close as possible to the ‘Bg_analyze software acceptance test description’ (IDC, 2005)	This requirement does not affect the design or the implementation of the software.
An installation manual shall be provided that follows the IDC Software Installation Plan Template (2003).	This requirement does not affect the design or the implementation of the software.
The installation plan document, and installation procedures described therein, should be as close as possible to the installation procedures described in the ‘National Data Centre Software Installation Plan’ (IDC, 2006).	This requirement does not affect the design or the implementation of the software.
It shall be possible to legally distribute the software to all States Parties.	The software does not use any components which would not allow for a legal distribution of the software to all State Parties.
It shall be possible to install the software at a National Data Centre (NDC).	It is possible to install the software at the NDCs if their computer infrastructure is compatible with the infrastructure of the IDC.

Requirement	Addressed by
The software shall not depend on third-party products that require a run-time license.	The software does not depend on third-party products that require a run-time license, apart from the Oracle database and Operating System.
The software shall allow for different efficiency equations, where each is defined by a code number between 1 and 99 (see CTBT/PTS/INF.96/Rev.6,Appendix I, §3.1. In this text the two equations mentioned here are given the codes 8 and 5 respectively). For each efficiency calibration one of these numbers (equations) should be selected and their corresponding parameters calculated. The efficiency curve was referred to by the pIDC in Arlington as the EER, the Efficiency vs. Energy Regression curve.	Not implemented.
The software shall prevent any user from deliberately or inadvertently changing any data in the Auto database.	The software can not protect the Auto database. This protection must be achieved on the level of database access rights.

Note: Additional requirements were identified and corresponding software changes designed and implemented in the test use of autoSaint.

APPENDIX II
CONFIGURATION PARAMETERS

Table 11 – *autoSaint* configuration parameters

Name	Type	Values range	Allowed in			Default value	Mandatory	Description
			Cmd line	DB	Default			
AVERAGEENERGYCALIBRATION	Boolean	YES, NO	Y	Y	Y	NO	N	If set, energy is based on average of energies calculated based on (0..N-1) and (1..N). If not set, energy is calculated based on (1..N)
BASELINEDIR	String	Up to 250 characters	Y	Y	N		Y	Baseline result directory (under RMS Home directory)
CAREATHRESHOLD	Float	Floating point number	Y	Y	Y	1000	N	Area threshold in the reference peak search for Calibration samples
COMPETITIONMAXENERGY	Integer	0-Max energy	Y	Y	Y	2000	N	High limit (in keV) for particulates competition search
COMPETITIONMINENERGY	Integer	0-Max energy	Y	Y	Y	100	N	Low limit (in keV) for particulates competition search
CONFIDENCELEVEL	Integer	0-100	Y	Y	Y	95	N	Calibration shift test confidence level (%)
DBDEFAULT	Boolean	YES, NO	Y	N	N		N	If set, *autoSaint* searches for the connect string file in the user's home directory
DBFILE	String	max file path length	Y	N	N		Y *	The FILE_NAME of the file containing the database connect string. *Either connect string, user/password/server or connect string file must be specified.
DBPASSWORD	String	max password length	Y	N	N		Y *	Database password. *Either connect string, user/password/server or connect string file must be specified.
DBSERVER	String	max server name length	Y	N	N		Y *	Database server name. *Either connect string, user/password/server or connect string file must be specified.

Name	Type	Values range	Allowed in			Default value	Mandatory	Description
			Cmd line	DB	Default			
DBSTRING	String	max database connection string length	Y	N	N		Y *	Database connect string. *Either connect string, user/password/server or connect string file must be specified.
DBUSER	String	max user name length	Y	N	N		Y *	Database username. *Either connect string, user/password/server or connect string file must be specified.
EFFICIENCYCOEFFS	Comma separated list of floats	Floating point numbers	Y	Y	N		N	Efficiency coefficients in the form c0,c1,...,cn/error
EFFICIENCYCALPOLYDEGREE	Integer	Integer number greater than 0	Y	N	Y	3	N	Efficiency Calibration Polynomial Degree
EFFVGSLPAIRS	Boolean	YES, NO	Y	Y	Y	YES	N	Use VGSL efficiency pairs, if available
EMPIRICALENERGYERRORFACTOR	Float	Floating point number	Y	Y	Y	0.5	N	Empirical energy error factor a of Error = centroid_energy_error + a * FWHM
EMPIRICALFWHMERRORFACTOR	Float	Floating point number	Y	Y	Y	0.01	N	Empirical FWHM error factor a of Error = centroid_fwhm_error + a * FWHM
ENERGY CALIBRATION COEFFS	Comma separated list of floats	Floating point numbers	Y	Y	N		N	Energy calibration coefficients in the form c0,c1,...,cn/error
ENERGYCAL POLYDEGREE	Integer	Integer number greater than 0	Y	N	Y	3	N	Energy Calibration Polynomial Degree
ENERGYCOEFFS	Comma separated list of floats	Floating point numbers	Y	Y	N		N	Energy coefficients in the form c0,c1,...,cn/error

Name	Type	Values range	Allowed in			Default value	Mandatory	Description
			Cmd line	DB	Default			
ENERGYIDTOLERANCEA	Float	Floating point number	Y	Y	Y	0.5	N	Empirical energy tolerance factor "a" in nuclides identification. Empirical tolerance = a+b*fwhm
ENERGYIDTOLERANCEB	Float	Floating point number	Y	Y	Y	0.0	N	Empirical energy tolerance factor "b" in nuclides identification. Empirical tolerance = a+b*fwhm
ENERGYWINNER	Enum	MRPA, MRPM, MRPQC, INPUT or INITIAL	Y	Y	N		N	Energy competition winner. One of MRP_A, MRP_M, MRP_{QC}, INPUT or INITIAL.
HELP	Boolean	YES, NO	Y	N	Y	NO	N	Help=YES to get this help
INTERMEDIATERESULTFILE	String	Up to 250 characters	Y	Y	N		N	Intermediate result file name
LOGLEVEL	Integer	0-9	Y	Y	Y	2	N	Log level (0-9)
MANUALBASELINE	String	Up to 250 characters	Y	Y	N		N	Path to manual baseline file
MANUALDB	String	Up to 250 characters	Y	Y	N		N	Manual Data Source
MANUALDBPASSWORD	String	Up to 250 characters	Y	Y	N		N	Manual DB Password
MANUALDBUSER	String	Up to 250 characters	Y	Y	N		N	Manual DB User
MINCOMPETITIONSCORE	Float	Floating point number	Y	Y	Y	0.1	N	Minimal plausible competition score
MIN CALIBRATION PEAKS	Integer	Integer number	Y	Y	Y	10	N	Minimal number of reference peaks needed for the calibration
NUCLIDDETECTABILITYTHRESHOLD	Float	Floating point number	Y	Y	Y	0.2	N	Detectability threshold in nuclide identification
OVERWRITE	Boolean	YES, NO	Y	Y	Y	NO	N	YES to overwrite existing results
QAREATHRESHOLD	Integer	NUMBER	Y	Y	Y	2500	N	Area threshold in the reference peak search for QC samples

Name	Type	Values range	Allowed in			Default value	Mandatory	Description
			Cmd line	DB	Default			
QCAIRVOLUME	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC air volume check
QCATIME	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC acquisition time check
QCCAT	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC auto category check
QCCOLLECTION GAPS	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC collection gaps check
QCCTIME	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC collection time check
QCDRIFT10D	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC 10 days drift check
QCDRFITMRP	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC MRP check
QCDTIME	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC decay time check
QCECR	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC ECR check
QCFLAGS	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC Ba-140_MDC and Be7_FWHM checks
QCFLOW	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC Flow check
QCFLOW500	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC flow 500 check
QCFLOWGAPS	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC flow gaps check
QCFLOWZERO	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC flow zero check
QCIDS	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC IDs check
QCPRELIMINARYSAMPLES	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC preliminary samples check
QCRTIME	Boolean	YES, NO	Y	Y	Y	YES	N	Enables QC reporting time check
REFLINETHRESHOLDA	Float	Floating point number	Y	Y	Y	1	N	Refline delta threshold coefficient a of a+be in the reference peak search
REFLINETHRESHOLDB	Float	Floating point number	Y	Y	Y	0.005	N	Refline delta threshold coefficient b of a+bc in the reference peak search
RESOLUTION CALIBRATION COEFFS	Comma separated list of floats	Floating point numbers	Y	Y	N		N	Resolution calibration coefficients in the form c0,c1,...,cn/error

Name	Type	Values range	Allowed in			Default value	Mandatory	Description
			Cmd line	DB	Default			
RESOLUTIONCAL POLYDEGREE	Integer	Integer number greater than 0	Y	N	Y	3	N	Resolution Calibration Polynomial Degree
RESOLUTIONCOEFFS	Comma separated list of floats	Floating point numbers	Y	Y	N		N	Resolution coefficients in the form c0,c1,...,cn/error
RESOLUTIONWINNER	Enum	MRPA, MRPM, MRPQC, INPUT or INITIAL	Y	Y	N		N	Resolution competition winner. One of MRP_A, MRP_M, MRP_{QC}, INPUT or INITIAL.
RISKLEVELINDEX	Integer	1-8	Y	Y	Y	3	N	Risk Level Index Index Risk Level k 1 0.000100 4.753420 2 0.000500 4.403940 3 0.001000 4.264890 4 0.010000 3.719020 5 0.050000 3.290530 6 0.100000 3.090230 7 1.000000 2.326350 8 5.000000 1.644850
RMSHOME	String	Up to 250 characters	Y	Y	N		Y	RMS Home Directory
SAMPLEID	Integer	Integer number	Y	N	N		Y	Sample ID
SAREATHRESHOLD	Integer	1000	Y	Y	Y	1000	N	Area threshold in the reference peak search for data samples
SCACDIR	String	Up to 250 characters	Y	Y	N		Y	SCAC result directory (under RMS Home directory)
SKIPCATEGORIZATION	Boolean	YES, NO	Y	Y	Y	NO	N	If set, categorization will be skipped
USEMRPAIRS	Boolean	YES, NO	Y	Y	Y	NO	N	If set, MRP parameters are recalculated from MRP pairs
VERSION	Boolean	YES, NO	Y	N	Y	NO	N	Version=YES to get the version

Name	Type	Values range	Allowed in			Default value	Mandatory	Description
			Cmd line	DB	Default			
XECOMPETITIONMAXENERGY	Integer	0-Max energy	Y	Y	Y	300	N	High limit (in keV) for Xenon competition search
XECOMPETITIONMINENERGY	Integer	0-Max energy	Y	Y	Y	25	N	Low limit (in keV) for Xenon competition search
XEGAMMAFACTOR	Float	Floating point number	Y	Y	Y	15.5188682	N	Xenon Gamma Factor
XESIGMAFACTOR	Float	Floating point number	Y	Y	Y	3.0	N	Sigma Factor

APPENDIX III
PARTICULATES PROCESSING SEQUENCE AS DEFINED IN TOR AND AS IMPLEMENTED

As Defined in TOR	As Implemented in *autoSaint*
Perform a set of actions (like checking analyst permissions, dumping info into standard input…)	Initialize logging, DB connection, load configuration. Set processing status to "A" Load sample data. Load MRPs.
	Calculate baseline.
Get initial processing parameters	
	Calculate LC. Calculate SCAC.
Run initial peak search	Run initial peak search.
Find reference peaks	Find reference peaks.
Update processing parameters using last output	Perform calibration using found reference peaks and perform competition.
	Calculate baseline.
	Write baseline to file.
	Calculate LC. Calculate SCAC.
	Write SCAC to file.
Run final peak search	Find peaks.
Reject peaks according to certain criteria	
Run Nuclide Identification routine	Run Nuclide Identification routine.
Calculate Minimum Detectable Concentrations (MDCs)	Calculate Activities and Minimum Detectable Concentrations (MDCs)
Run categorization routine	(Optional, currently unused) Perform categorization.
Populate Data Base with analysis results	
Run Quality Control program and write into files	Run Quality Control.

As Defined in TOR	As Implemented in *autoSaint*
	Set processing status to “P”

XENON PROCESSING SEQUENCE AS DEFINED IN TOR AND AS IMPLEMENTED

As Defined in TOR	As Implemented in *autoSaint*
Perform a set of actions (like checking analyst permissions, dumping info into standard input, fault check…)	Initialize logging, DB connection, load configuration. Set processing status to "A" Load sample data. Load MRPs.
Calculate BASELINE	Calculate baseline for main and preliminary samples.
Get initial processing parameters	
Calculate LCC/SCAC	Calculate LC for main and preliminary samples. Calculate SCAC for main and preliminary samples.
Run initial peak search	Run initial peak search.
Find reference peaks	Find reference peaks.
Update processing parameters using last output	Perform calibration using found reference peaks and perform competition.
Re-calculate BASELINE	Calculate baseline for main and preliminary samples.
Store BASELINE	Write baseline to file.
Re-calculate LCC/SCAC	Calculate LC for main and preliminary samples. Calculate SCAC for main and preliminary samples.
Store SCAC	Write SCAC to file.
	Run method 1 for Xe-isotopes quantifications
	Run method 2 for Xe-isotopes quantifications
Calculate Activities	Calculate Xe-Isotopes Activities
Calculate MDAs/MDCs	Calculate Xe-Isotopes MDAs/MDCs
	(Optional, currently unused) Perform categorization.

As Defined in TOR	As Implemented in *autoSaint*
Run Quality Control program	Run Quality Control.
	Set processing status to “P”.

APPENDIX IV
ABBREVIATIONS

ANSI	American National Standards Institute
ARR	Automatic Radionuclide Report
CTBTO	Comprehensive Nuclear-Test-Ban Treaty Organization
CCF	Coincidence Correction Factor
DCF	Decay Correction Factor
ECR	Energy Channel Regression
FWHMC	Full Width at Half Maximum in Channels
GUI	Graphical User Interface
gODBC	gbase Open Database Connectivity
IDC	International Data Centre
IEC	International Electrotechnical Commission
ISO	International Standard Organization
LCC	Critical Level Curve
MDA	Minimum Detectable Activity
MDC	Minimum Detectable Concentration
MRP	Most Recent Prior
NDC	National Data Centre
ODBC	Open Database Connectivity
PTS	Provisional Technical Secretariat
QC	Quality Control
RER	Resolution Energy Regression
RRR	Reviewed Radionuclide Report
SAINT	Simulation Assisted Interactive Nuclear Review Tool
SCAC	Single Channel Analyzer Curve
SDD	Software Design Document
SQL	Structured Query Language
TOR	Terms Of Reference

REFERENCES

[IDC_CS_2002]	International Data Centre (2002). IDC Software Coding Standard
[AUTO_SAINT_SRS]	Auto-SAINT Software Requirements Specification. IDC/auto/saint/SRS, 2003-07-20
[AUTO_XE_SAINT_SRS]	Auto-Xe-SAINT Software Requirements Specification. IDC/auto_Xe_saint/SRS, 2007-06-15
[IDC_SYSLOG_2003]	Using Syslog at CTBTO' (IDC, 2003)
[AUTO_SAINT_QP]	Radionuclide Software Development Quality Plan, AWST-TR-07/01, version 1.0, 2007-01-21
[IDC_SUT_2003]	Software User Tutorial Template, IDC/TBD1/SUT, 2003-05-27
[BG_ANALYZE_SUT]	Bg_analyze Software User Tutorial, IDC/bg_analyze/SUT, 2005-02-05